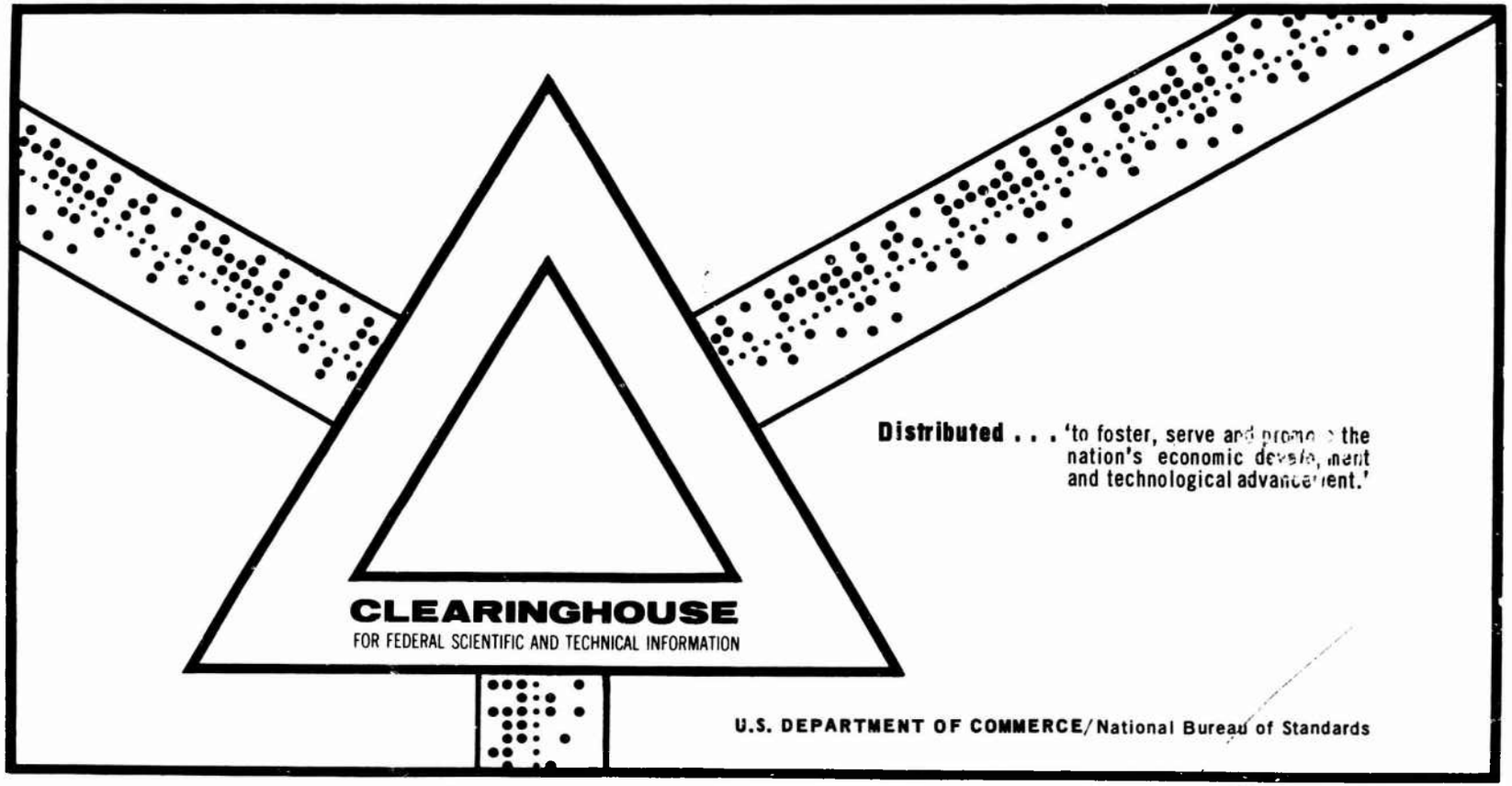
STUDY FOR APPLYING COMPUTER-GENERATED IMAGES TO VISUAL SIMULATION

Robert A. Schumacker, et al

General Electric Company
Daytona Beach, Florida

September 1969



AFHRL-TR-69-14

SEPTEMBER 1969

AIR FORCE HUMAN RESOURCES LABORATORY

STUDY FOR APPLYING COMPUTER-GENERATED IMAGES TO VISUAL SIMULATION

R. Schumacher
B. Brand
M. Gilliland
W. Sharp
General Electric Company

Sponsored by
Training Research Division
Wright-Patterson Air Force Base, Ohio

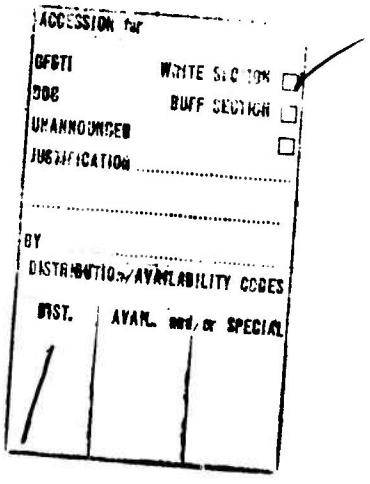
AIR FORCE SYSTEMS COMMAND

BROOKS AIR FORCE BASE, TEXAS

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This document has been approved for public release and sale; its distribution is unlimited.



Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

800 - December 1969 - CO455 - 104-2286

AFHRL-TR-69-14

SEPTEMBER 1969

STUDY FOR APPLYING COMPUTER-GENERATED IMAGES TO VISUAL SIMULATION

R. Schumacher
B. Brand
M. Gilliland
W. Sharp
General Electric Company

Sponsored by
Training Research Division
Wright-Patterson Air Force Base, Ohio

FOREWORD

This is the Final Technical Report on a Study for Applying Computer-generated Images to Visual Simulation, conducted for the Air Force Human Resources Laboratory (AFSC), USAF, under Contract No. F33615-69-C-1280.

The purpose of the study was to determine the application of computer-generated images to visual simulation for pilot training. The work was conducted during the time period January 1969 - July 1969 by the General Electric Electronics Laboratory, Syracuse, New York, under the sponsorship of the Company's Apollo Systems Department, Daytona Beach, Florida.

The Air Force program monitor is Mr. James Basinger.

This report was submitted by the authors, July 1969.

This technical report has been reviewed and is approved.

Gordon A. Eckstrand, Ph.D.
Chief
Training Research Division
Air Force Human Resources Laboratory

ABSTRACT

This report describes the results of a system design study for applying digital image generation techniques to visual simulation for pilot training. The computer-generated images are to provide out-the-window scenes for a flight simulator which is to be used for training Air Force pilots.

No existing visual system can provide all of the capabilities which are desired in a flight simulator. Digitally generated scenes do overcome many of the shortcomings associated with more conventional approaches but have had limited application because of the difficulty of computing enough image detail. The ability to generate images of more complex and realistic environments is closely tied to advances in digital device technology. The study assesses the impact of recent developments in this area on the design of an image generating system.

The conceptual design of an image generator is described. The principles of operation, the system configuration, and operational characteristics are discussed. Several key problem areas are explored in depth. Feasible methods of implementation with presently available hardware are examined and an estimate of the hardware complexity is given.

SUMMARY AND CONCLUSIONS

PROBLEM

Digital computer-generated images have been used in visual simulation for research studies. Due to recent advances in computer and integrated circuit technology, higher-quality images can be generated, and are suitable for visual simulation for pilot training. The image generator must be defined before equipment design is initiated.

APPROACH

A study was initiated to explore technology for a feasible digital image generator. The study was to outline the operation of the image generator, define the required subsystems and describe the required "off-the-shelf" equipment.

RESULTS

This report describes an image generator for pilot training. This generator used a general-purpose computer with three special computer processors for object, terrain and point light source generation. The output of this generator is compatible with standard television equipment.

CONCLUSIONS

A digital image generator for pilot training is feasible. The equipment design is complex but is also extremely flexible and should be very reliable.

James D. Basinger
Project Engineer
A. F. Human Res. Lab.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (concluded)

LIST OF ILLUSTRATIONS

List of Illustrations (continued)

List of Illustrations (concluded)

LIST OF TABLES

I. INTRODUCTION

A. PURPOSE OF STUDY

This report describes the results of a system design study for applying digital image generation techniques to visual simulation for pilot training. The computer-generated images will provide out-the-window scenes for a flight simulator that is to be used for training Air Force pilots.

No existing visual system can provide all of the capabilities that are desired in a flight simulator. Conventional image generators that use film or physical models have limited dynamic freedom, constrained operating envelopes, and an image quality that is limited by a finite depth of field. These problems do not arise in a computer-generated display. The images possess an infinite depth of field, follow all dynamic inputs, maintain proper perspective, and the models, consisting only of numbers in a computer memory, are easily modified or exchanged for others. These features have been enjoyed by engineering and research groups who accepted a somewhat symbolic representation of the environment in exchange for flexibility and dynamic fidelity not attainable in other systems. However, successful application of this technology to pilot training simulators requires that sufficient environment detail and realism be provided to immerse the trainee visually in his tasks.

The ability to generate images of more complex and realistic environments is closely tied to advances in digital device technology. The study assesses the impact of recent developments in this area on the design of an image-generating system. The organizational concepts that now appear attractive differ considerably from those used in the past. The purpose of the study is to evaluate these concepts relative to the pilot training tasks, to define an image-generating system that implements these concepts, and then to describe its capabilities.

B. SCOPE AND APPROACH

The study is concerned primarily with the image-generation portion of a flight simulator. The display device and flight dynamics computer are considered only to the extent necessary to specify the interfaces. The image generator is self-contained and is driven by a flight dynamics computer that provides position and attitude information for the moving vehicles. Images are generated for two separate views and video and synchronizing signals are provided for television monitors. Each frame of the scene is generated in real-time, in response to the dynamic inputs, from an internally stored numerical description of the environment.

A brief requirements analysis is included to show how the image-generating techniques are to be applied to provide environments suitable to the training tasks. Only those characteristics that influenced the basic design of the system were considered.

The image-generating system is not designed around a specific application, but represents what might be called a basic system. As such, it provides a vehicle for the study that demonstrates the application of several image-generating techniques to the problems of pilot training. The capabilities of the system are thought to be typical of those that might be needed in a specific application, but are by no means maximal. One of the principal objectives of the study was to devise a flexible system configuration wherein the image-generating capability could be modularly expanded or contracted as dictated by future requirements.

The image generator is described on a block-diagram level. Its basic configuration was established early in the study and the remainder of the effort was devoted to the conceptual design of the resulting subsystems. Three topics were given special emphasis, because of their significant impact on the system. These include:

1) priority, or hidden-line, problem
2) object processing section
3) surface generation

The first two topics are concerned with the generation of three-dimensional objects. The object-processing section is organized around a list-processing concept for providing priority among objects. This concept significantly reduces the hardware and software involved in object generation and allows for future expansion of capability with only linear increases in complexity. This new approach to the problem requires that certain restrictions be placed on the environmental model. A large portion of the study was devoted to deriving these restrictions and evaluating their impact on environment modeling.

The object-processing section comprises the bulk of the special-purpose hardware in the system. Its organization is intimately involved with the practical aspects of its implementation, particularly in view of its probable implementation with large-scale arrays. In all of these areas, the study often probed to considerable depth, in order to reach conclusions at the block-diagram level.

C. RESULTS

The conceptual design of the image generator is described in this report. The basic principles of operation, the system configuration, and its operational characteristics are summarized in Section III. Section IV describes the priority problem, the proposed solution, and the derivation of the required geometric restrictions. The following section discusses pertinent topics concerning the state-of-the-art in the semiconductor industry. Present and near-future capabilities are reviewed and a basis is established for hardware estimation.

Sections VI through X describe individual sections of the image generator in more detail. Finally, an estimate of the hardware complexity of the system is given, in Section XI, in terms of the approximate number of circuit cards and their logic complexity.

II. SYSTEM REQUIREMENTS

A. CHARACTERISTICS OF COMPUTED IMAGES

Some of the basic properties of computed images are stated here to illustrate the types of scenes that may be generated. The environment model is generally processed by three separate image-generating techniques, each of which is particularly suited to provide certain types of environment information. These techniques include:

1) object generation
2) surface generation
3) point-source generation

In each case, the image is computed in real time and mathematically correct perspective is maintained.

Figure 1 is a photograph of one of the displays of the Electronic Scene Generator installed at the NASA Manned Spacecraft Center, Houston, Texas (Contracts NAS 9-1375 and NAS 9-3916). It illustrates the application of the three image-generating techniques.

Object generation is used to form the command and service module, which is capable of six degrees of motion. The vehicle is approximated by convex planar faces, which are bounded by straight edges. The object generating capacity of a system is specified by the number of edges that can be used to model the environment. In this case, the entire 240-edge capacity of the system was used to model the space vehicle. Much of the detail is not visible from this attitude and distance. A color is assigned to each face as part of the numerical description of the object. Although the colors are usually fixed for a given mission, the capability exists for changing the color or intensity from frame to frame.

Although the object appears to be solid, its faces have no thickness. The observer is free to move arbitrarily close to a face, or through it, in which case the inside surface would be seen.

The object-generating technique is the most general of the three and, in fact, could be used to provide all of the scene information, if one had no limit on the number of edges available. Much of the environment, however, is essentially two-dimensional and can be generated by more economical means.

The ground plane in the photograph is formed by surface-generating techniques. It consists of an unbounded plane surface, which is covered by texture patterns. Only one of four nested patterns can be seen from this altitude; the others become visible at progressively lower altitudes. By nesting the patterns in this manner, a useable level of detail is always provided over

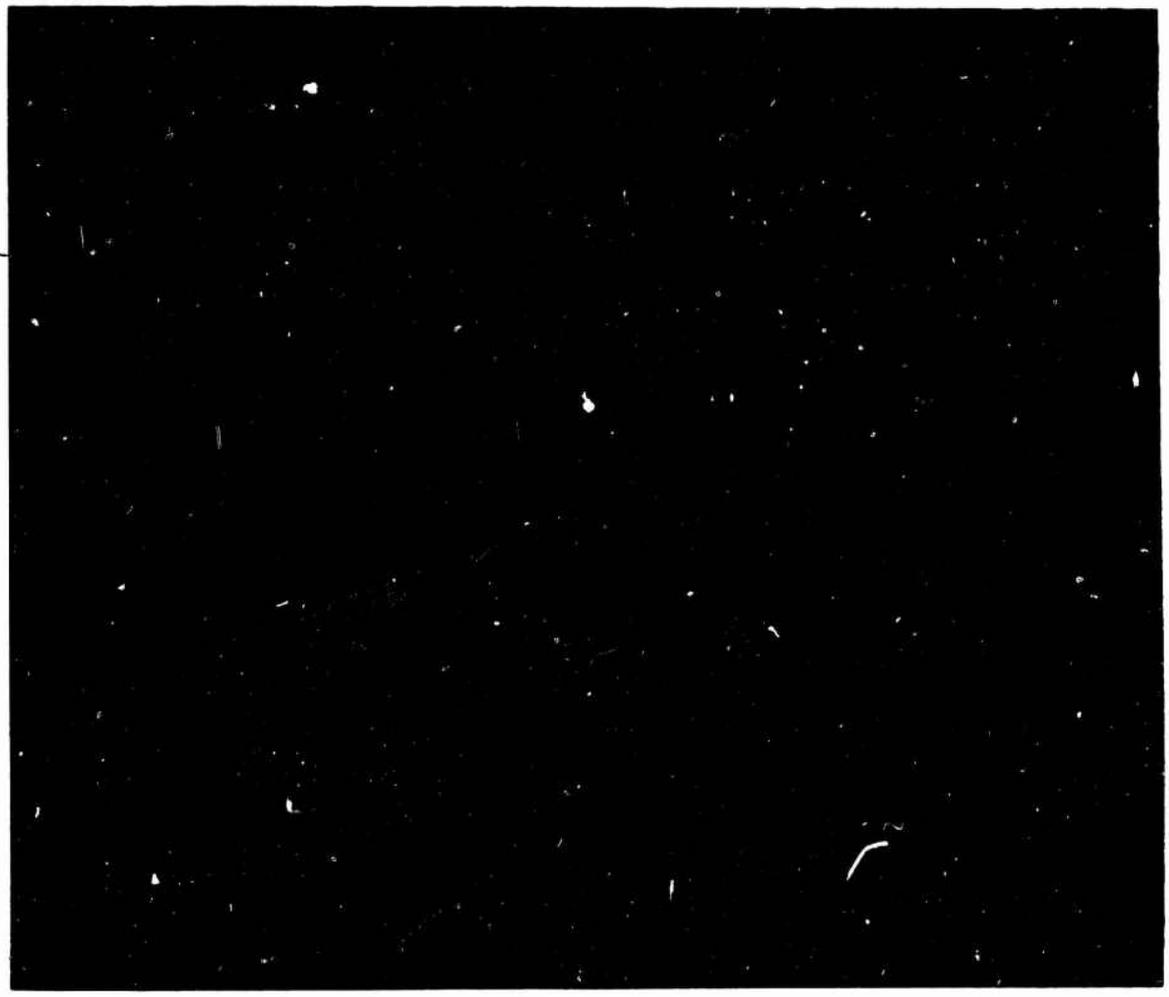


Figure 1. Example of Scene Generated by Basic Techniques

a large dynamic range. The apparent curvature of the horizon is a special effect provided for this simulation and does not indicate that the surface is curved.

The texture patterns are stored in "maps", which are addressed by the ground coordinates of the scanning ray. Each of the patterns repeats along both cardinal axes. An extensive network of patterns is thus provided with relatively little storage. The stored patterns are easily changed, but must retain their rectilinear format because they are composed of square cells.

Point-source techniques are illustrated by the stars in the picture. This approach is useful for portraying objects whose image size is small relative to the resolution of the raster. The point sources are displayed at a constant size, usually one raster element by one raster line pair.

The technique is applicable to the generation of lights when changes in image size with range can be neglected. Although a close approach to an extended light source cannot be correctly represented by this technique, the constant size representation does prevent image interaction with the raster structure which, at larger ranges, would result in intermittent visibility.

B. OPERATIONAL REQUIREMENTS

The image-generating capability of the system was specified in terms of minimum scene content and also by a number of operational requirements. The environment is to be composed of both planar information for ground and sky and three-dimensional objects such as buildings and aircraft. The two views may be used separately in different cockpits, or together to provide two views for a single pilot. Therefore, each of the views must be capable of the maximum complexity. A minimum object-generating capability of 500 edges is specified for each view. The following operational tasks must be accommodated:

1) Circling approach, landing and take-off, day and night
2) Taxiing, day and night
3) Formation flying
4) Air-to-air combat
5) Aerobatic maneuvers
6) Air-to-ground weapons delivery

1. Airport Operations

The first two training tasks concern operation in the vicinity of an airport. Most of the object generating capability would likely be devoted to the airport itself for depicting taxiways, ramps, runway, runway markings, building on the airfield, and nearby obstructions. Secondarily, object generation may be used to show prominent landmarks and gross geographical features which would serve to orient the pilot when the airport was not in sight. Most features of this sort could be limited to a ten-nautical-mile radius of the airport, the area within which approaches are normally conducted.

Figure 2 shows a very simple airport environment, which was generated by the NASA system. The runway markings do not conform to standards and the open hangars are probably unnecessarily detailed. Object-generating techniques were used exclusively with only a few edges devoted to defining the ground plane. Experience with this model has shown that once the airport is out of sight, there are very few visual cues that the pilot can use to judge his position, altitude and velocity, demonstrating the importance of having surface texture.

It is estimated that 500 edges can be used to model an airport with two marked runways, principal taxiways, several buildings, and modest surrounding terrain. It is assumed that no other moving aircraft are to be depicted for these training tasks.

The most demanding image-generating task is encountered at dusk, when both lights and objects are visible. It is assumed that lights of interest are those associated with the active runway, its approach path and certain taxiways. A 10,000 foot runway with lights every 100 feet would require 200 lights. Limited taxiway lighting would probably require another 200 lights. With approach lights, runway end identifier lights, and a few lights for marking obstructions, the total requirement comes to approximately 500. The only practical way of generating this number of lights is by point-source techniques. The fixed size of the point-source images will result in improper perception of the size of the lights when the pilot is on the ground. If a 1000-line display covers a 60-degree field of view, then its elements subtend approximately three arc-minutes. A six-inch runway light viewed from the center of a 150-foot runway might actually subtend as much as six display elements instead of one. Although this error is large at close ranges, it does not appear during the approach phase, where the image size would normally be smaller than the display resolution.

If the system is used in conjunction with a color television monitor, the color of the lights may be chosen with the same flexibility as are the object colors. Most lights on an airport are either red, blue or white. The ability to change the intensity of the lights is probably important to maintain proper illumination balance in going from dusk to full night operation.

The image-generating system places no restrictions on aircraft maneuvers. The pilot has complete freedom to execute a missed approach, select an alternate runway — or crash!

2. Other Aircraft

The requirements for formation flying and air-to-air combat are quite similar. In both cases, the primary visual cue is the other aircraft. In the latter case, the other aircraft may not be in sight as often as desired, and terrain features become important for orientation. Both training tasks may range over large areas of terrain.

It is assumed that, in each case, one other aircraft is involved. In air-to-air combat, the two views would be in separate cockpits. The same configuration could serve for formation flying if an active lead aircraft were

used; otherwise, both views might be provided for one pilot and the lead aircraft could be placed on a preprogrammed flight path or put under the control of an instructor. In either case, both views would be operating in the same environment. Because moving objects are involved in these environments, priority considerations require certain restrictions on the relative positions of the two aircraft and any three-dimensional terrain. If all maneuvers are conducted at altitudes higher than the terrain, then the problem can be ignored. This subject is discussed further in Section IV.

Figure 3 is an artist's rendition of a relatively complex model of an F-105. Approximately 150 edges are used. If this level of detail were used to model each of the aircraft involved in air-to-air combat, then 70 percent of the object-generating capability would be available for terrain modelling.

3. Aerobatic Maneuvers

No particular problems are posed by this task, since the image generator is capable of responding to any new set of inputs on a frame-by-frame basis. Any of the environment models previously discussed could be used here. Recognizable terrain, unique landmarks, and textured surfaces are features that would be appropriate.

4. Air-to-Ground Weapons Delivery

The significant problems that are added in this task are: (1) the ballistics of the weapon, (2) portrayal of the weapon during flight, and (3) portrayal of impact effects. The ballistic computations become quite complex for many weapons. The available computing time in the general-purpose computer would limit the fidelity of this computation if it were performed by the image-generating system. An alternative approach could be employed if sophisticated ballistics were required. The dynamics of the weapon could be computed by the flight dynamics computer and supplied as inputs through the channel otherwise used to control a second aircraft.

Either point or object generating techniques could be employed to depict the weapon in flight. Tracer-bullet paths would best be provided by the former; air-to-ground missiles would be more accurately represented by the latter technique. Weapons effects may be shown symbolically in a variety of ways, such as changing the object's color, enlarging the object, or eliminating it. The capabilities of object and point source techniques determine the kind of effects that can be shown; the amount of reserve computing power determines the degree of sophistication that can be employed.

C. SUMMARY OF REQUIRED CAPABILITIES

The following image-generator capabilities are needed to satisfy both the stated requirements and those implied by the training tasks:

1) Each view will have a 500-edge object-generating capability. This implies that, when both views are used in a single cockpit, they will each view the same 500-edge environment. When the views are used in separate cockpits, such as for air-to-air combat, they will be

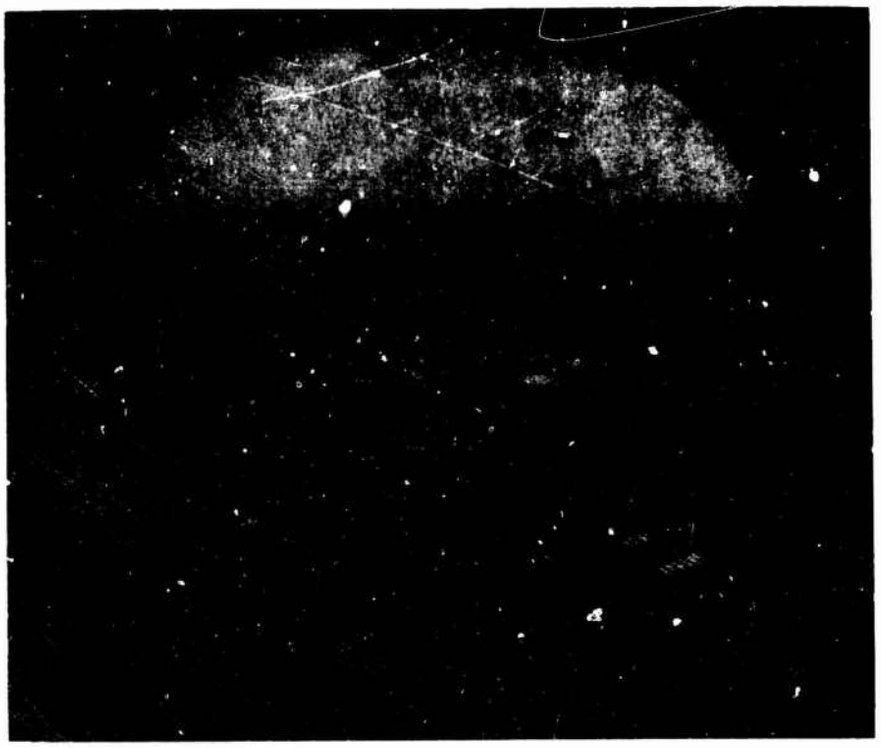


Figure 2. 240-Edge Airport Scene

Figure 3. Artist's Drawing of F-105

operating in a common environment, except that each may view a different aircraft. Thus, the terrain portions of the environment will be the same, but storage must be provided for two separate aircraft. Assuming that each aircraft is modeled with 150 edges and the remaining capability is used for terrain, environment storage must be provided for 650-edge complexity.

2) Each view will have a textured surface generator that will provide images of one plane surface. The primary use of textured surfaces will be to provide motion cues over large expanses of ground. At the present time, insufficient data are available to evaluate the efficiency of using this technique to provide recognizable features such as runways and roads. Therefore, its images will be principally symbolic. However, rather than having all patterns repeat indefinitely, maps will be provided to select semi-unique locations for certain patterns as an aid to navigation. A new approach to surface generation will be used (rather than that used in the NASA system) so that standard television monitors may be employed.

3) Point-source generators will provide the capability of displaying at least 500 light-source images. It is assumed that only one aircraft will be flying in an environment which uses the full point-source generating capability. This would permit both views to be used by one pilot during this training task but would not allow two independent simulations. If the two views are used together, the point sources may appear in either of them and in any combination. All fixed airport lights will be generated by this technique. The color (or gray shade) of the lights will be represented by a six-bit number in the same manner as object faces.

4) Sufficient reserve computing capability will be provided to process modest air-to-ground weapons effects. This would include one or two missiles in flight at the same time, or tracer fire. In the latter case, the capabilities of the point-source generators would be devoted to this task.

5) Sky will be represented by a solid "background" color. The color may be any of those normally used to represent objects.

III. SYSTEM DESCRIPTION

A. PRINCIPLES OF OPERATION

This section of the report describes the basic mathematical principles employed in the three image-generation techniques, shows the computational tasks that must be performed, and establishes nomenclature for later reference.

1. Object Generation

The object generating technique is concerned with finding the perspective image of the three-dimensional portion of the environment. The environment features are approximated by convex polygons arranged to form objects. There are two main facets to object generation. The first is to find the display plane images of the individual polygons. The second is to delete those portions of the polygons that are hidden by others. This section is primarily concerned with the first topic. The obscuration problem is discussed in Section IV.

The perspective images are computed on a display plane that is a mathematical analog of the actual display device - usually a cathode ray tube. The display plane is assumed to be flat and, for purposes of this discussion, is square in format. The display plane is mathematically scanned in the same manner as the electron beam traces out the raster pattern in the CRT. Table I lists and defines the quantities used in describing the display plane and the raster scan depicted in Figure 4. The observer's eye position, called the station point, is the origin of a display coordinate system with axes u, v, and w.

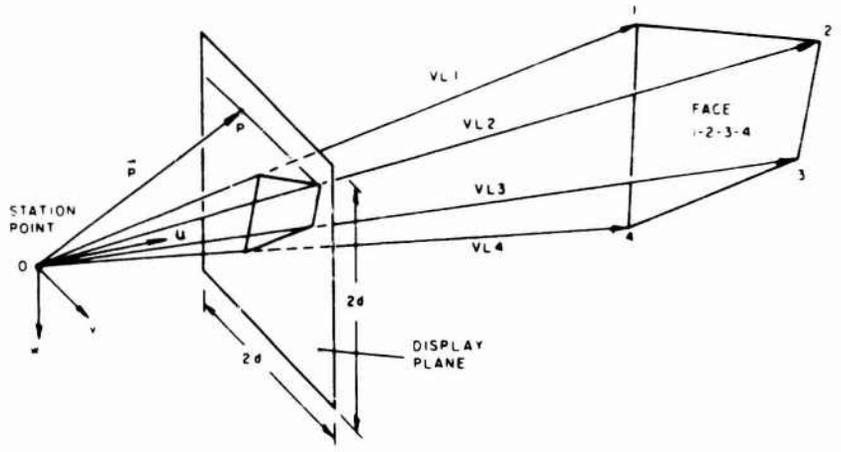


Figure 4. Image Formation

TABLE I

DISPLAY QUANTITY NOTATION

1.	I	- line number (within a field)
2.	J	- element number
3.	d	- display half width and half height
4.	p	- distance from station point to display plane center
5.	m	- number of active display elements
6.	n	- number of active display lines
7.	A	- edge intercept quantity, J(I=0).
8.	B	- edge slope, $\Delta J/\Delta I$
9.	$\vec{PC}$	- vector from station point to upper-left-hand corner of display expressed relative to d: $PC = (p/d)\,\vec{u} - \vec{v} - \vec{w}$
10.	$\vec{PL}$	- vector in the line direction = $(4/n)\vec{w}$
11.	$\vec{PE}$	- vector in the element direction = $-(2/m)\vec{v}$

The display plane is centered on the u axis and perpendicular to it. Axes v and w are oriented, respectively, along a raster line in the direction of increasing element number, and perpendicular to the lines in the direction of increasing line number.

The raster lines are numbered from the top of the display to the bottom, and elements are counted from left to right. The raster pattern is traced by the vector P, which emanates from the station point and points to successive display elements. This vector is described in terms of its three components, PC, PL and PE. PC locates the starting point of the scan in the upper left-hand corner. The vector PE is used to step P along the scan line, and PL is used to go from line to line. The P vector to any point on the display plane can be expressed in terms of an element number, a line number and these three components. These quantities have been defined in a form that is convenient for computation and are normalized to the display width so that only the u component of PC need by changed to modify the computed field of view.

Figure 4 shows a simple three-dimensional situation which illustrates the principles of image formation. The P vector and the vectors to the vertices of the face are the only quantities needed to find the image. It is necessary that all of these vectors be expressed in a common coordinate system. In practice, the vertex coordinates are stored in an environment memory relative to some object coordinate system. An L vector is computed for each

frame which relates the station point and the object coordinate system; the VL vectors are then easily formed by addition. It is, therefore, only necessary to transform the three P vector components and the L vector to object coordinates to provide compatibility.

In order to find the image of the face shown, we must know when the extended P vector pierces the face 1-2-3-4. The problem is further simplified by examining the position of the scanning ray relative to the edges. In these terms, the face is pierced only when the ray is simultaneously below edge 1-2, to the left of edge 2-3, above edge 3-4, and to the right of edge 4-1. The edges are thought of here as infinite lines rather than line segments.

Consider the edge 1-2 and its image. The scanning spot is below edge 1-2 only when the scalar triple product $\vec{P} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2}) > 0$. Call this condition F_1. Similarly,

P left of 2-3 is equivalent to $\vec{P} \cdot (\overrightarrow{VL2} \times \overrightarrow{VL3}) > 0$ and is called F_2

P above 3-4 is equivalent to $\vec{P} \cdot (\overrightarrow{VL3} \times \overrightarrow{VL4}) > 0$ and is called F_3

P right of 4-1 is equivalent to $\vec{P} \cdot (\overrightarrow{VL4} \times \overrightarrow{VL1}) > 0$ and is called F_4

Therefore, the condition that the P vector be inside of the image of face 1-2-3-4 is $F = F_1F_2F_3F_4$, where juxtaposition denotes logical AND.

If the side of the face where the vertices appear in clockwise order, as above, is called the front face, then F is the condition for drawing the front face. Call 4-3-2-1 the back face, and say that B is the condition that the extended P vector ray pierces the face through the back side. Then $B = \overline{F}_1\overline{F}_2\overline{F}_3\overline{F}_4$, where the bar denotes logical negation. By convention we will say that the back side of a face is that side interior to a polyhedron.

For a given edge, say 1-2, we must monitor the sign of $Q = \vec{P} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2})$. $Q < 0$ indicates that the scanning ray is on one side of the edge; $Q > 0$ indicates that the scanning ray is on the other side. Alternatively, we may look for $Q = 0$, knowing the direction (positive or negative) in which Q is changing. The scanning spot, defined by the P vector, describes a raster pattern moving across successive elements on each consecutive line. Q may be expressed in terms of the raster parameters by stating the P vector in terms of its starting location, PC, and its line and element dependent components PL and PE respectively. Expressing P in terms of line and element numbers:

$$\vec{P} = \overrightarrow{PC} + I\ \overrightarrow{PL} - J\ \overrightarrow{PE}$$

Then: $$Q = \vec{P} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2}) = \overrightarrow{PC} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2}) + I\ \overrightarrow{PL} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2}) - J\ \overrightarrow{PE} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2})$$

$$Q = QC + I\,QL - J\,QE$$

where

$$QC = \overrightarrow{PC} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2})$$

$$QL = \overrightarrow{PL} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2})$$

$$QE = \overrightarrow{PE} \cdot (\overrightarrow{VL1} \times \overrightarrow{VL2})$$

Suppose that QE = 0. Then Q = QC + IQL changes at most each line period. Its sign can be checked at the beginning of each raster line. This situation arises when an edge image is exactly parallel to a raster line.

Suppose that $QE \neq 0$. Then $Q = QC + IQL - J\,QE = 0$ when $J = J_o = QC/QE + I\,QL/QE$. J_o is the element number for the particular line I at which the scanning ray intersects the edge. The direction in which Q is changing as a scan line is traversed is given by the sign of QE.

Q crosses zero when the scanning vector, P, comes into the plane containing the two vectors VL1 and VL2. Since we are interested only in the zero crossing, the magnitude of the scalar triple product may be scaled arbitrarily. This allows the VL and P vectors to be independently multiplied by non-zero scalars. In practice, the VL vectors are normalized and the P vector is expressed relative to the display dimension "d".

The equation of the edge image is:

$$J_o = A + IB$$

where

$$A = \frac{QC}{QE}$$

$$B = \frac{QL}{QE}$$

When B is very large, J_o is not of interest because it changes radically from line to line. This situation corresponds to an edge image which is nearly parallel to a raster line.

We will compute J_o only for those cases where the edge image crosses two or more raster lines; otherwise we will compute the line number which satisfies the equation:

$$I_o = -\frac{QC}{QL}$$

The range of the numbers which must be handled is thereby limited.

Thus two situations arise. For the normal case, the edge image is represented by:

$$A = \frac{QC}{QE}$$

$$B = \frac{QL}{QE}$$

When the image is almost parallel to a raster line, the A and B quantities have a different interpretation;

$$A' = \frac{QC}{QL}$$

$$B' = 1.$$

Here A' is the negative of the line number and B' is a number which, when added to A', will change the sign of the sum when the image line is reached.

In addition to the A and B quantities, the sign of QE (for the normal case), or the sign of QL (for the parallel case) is required so that the sign of Q may be determined from the sign of A, that is, to indicate the side of the edge we are on for the zeroeth element of each line.

There is one ambiguous case, which can arise because edge image quantities are formed for edges that lie behind the station point as well as for those which lie in front of it. The ambiguity can be eliminated by testing the station point position to see if it lies on the front or back of the face. If the station point is on the front side, then is it impossible to see the back. This test, which must be performed for every face in the environment, is called the "aspect" test.

The computation of the A and B edge parameters involves a large amount of repetitious vector arithmetic. Even the most powerful general-purpose computers of today could not complete these computations on a meaningful number of edges within a frame time. For this reason, special-purpose arithmetic units are employed to implement the Object Computing Section. It should be noted that the vector cross-product computations do not depend on the attitude of the display plane. As a consequence, these results may be used to compute a number of different views from the same station point.

2. Surface Generation

The techniques of surface generation are used to compute the image of a single plane surface. The approach used is somewhat the inverse of that used for computing objects. In the latter case the objects were projected onto the display plane and their image was determined relative to the raster elements. In surface generation, on the other hand, the raster elements are projected onto the plane surface.

The geometry of the surface scanning problem is shown in Figure 5. The coordinates of the plane surface are defined by the (X, Y, Z) system.

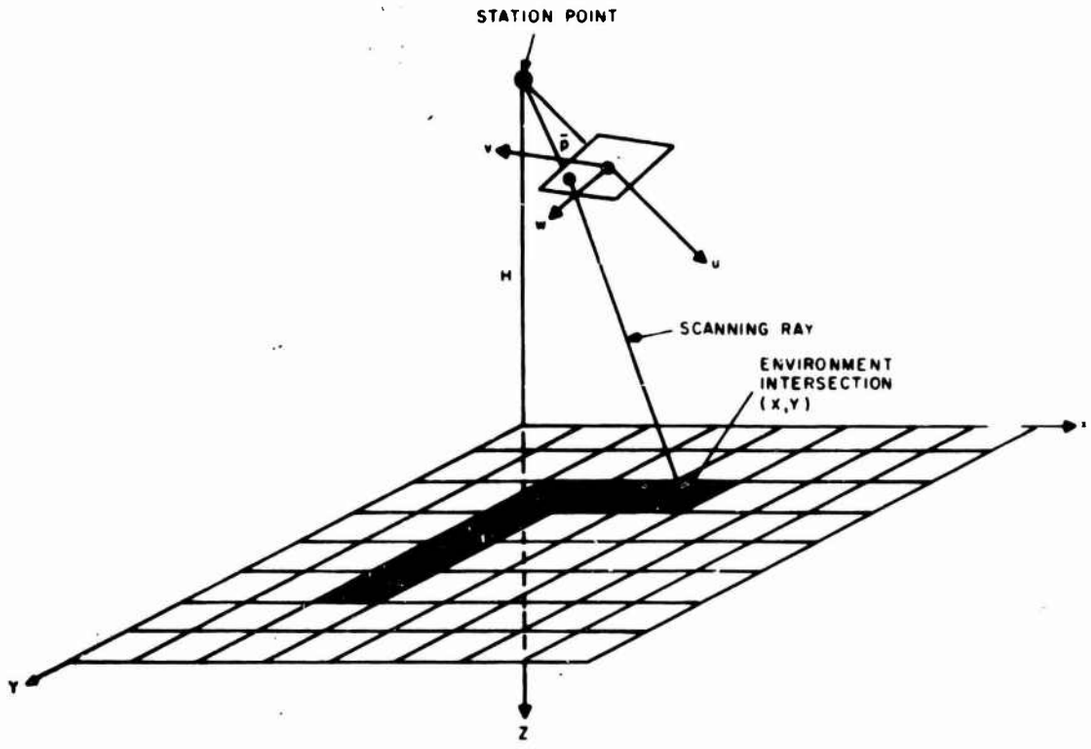


Figure 5. Surface Generation Geometry

The station point is assumed to be at an altitude H above the plane. For simplicity, the station point is shown directly over the origin of the ground coordinate system. The coordinates of the intersection of the extended P vector and the plane surface must be found.

With the P vector expressed in ground coordinates, the X and Y intersection points may be found by proportion. They are:

$$X = \frac{H\,P_x}{P_z}$$

$$Y = \frac{H\,P_y}{P_z}$$

The subscripts denote the components of the vector. The intersection points must be computed for each element on the display plane. By expressing the P vector in terms of the line and element numbers, the above expressions may be rewritten to show this dependence explicitly:

$$X = \frac{H(PC + IPL + JPE)_x}{(PC + IPL + JPE)_z}$$

$$Y = \frac{H(PC + IPL + JPE)_y}{(PC + IPL + JPE)_z}$$

In the general case, both the numerator and denominator of these expressions change from element to element. This would imply that two divisions must be performed to compute the intersection coordinates for each element. This problem was circumvented in previous implementations by using a special display with a rolled raster. By properly rolling the mathematical display plane (and using the physical display to re-introduce the roll) it is possible to cause the Z component of PE to vanish, making the denominator constant over a raster line. The ground coordinates are then linear functions of the element number and may be found by addition. There are difficulties associated with the requirement for a special display of this type, particularly when working with color. This study considers an approach to surface scanning that provides the quotients by taking advantage of the fact that only certain changes in the results are significant. The details of this method are discussed in Section IX.

The ground plane information is stored in "maps" or digital memory that can be addressed by the intersection coordinates. Patterns are defined by storing colors in memory locations that correspond to square cells on the ground plane. The patterns may be made to repeat both in X and Y by ignoring higher-order bits in the address. A hierarchy of patterns is provided by using a number of different maps, each with a different basic cell size or scale factor. Maps may also be used to specify the location of patterns generated by other maps. The primary advantage of the surface technique is that once a pattern is defined by a map, it may be used repetitiously over a surface with no additional hardware.

3. Point-source Generation

The point-source technique, conceptually the simplest of the three techniques, is used to find the display plane image of a set of points in the environment. All that need be determined for each point is the element and line number of the nearest display plane element and the color to be used.

The computation of point-source images is most conveniently performed with all vectors expressed in display coordinates. If the location of a point source is specified relative to the station point by a vector S, then its image coordinates are given by:

$$I = \frac{n}{2}\left[1 + \frac{p}{d}\frac{S_w}{S_u}\right]$$

$$J = \frac{m}{2}\left[1 + \frac{p}{d}\frac{S_v}{S_u}\right]$$

The line and element number must be computed to the nearest integer. If intense point sources are required, it may be necessary to display them on both fields of the raster (and occupy two lines) to avoid flicker. In this case, it is necessary to retain one fractional bit of the line number, so that the proper line may be selected during each field.

Unlike objects, point-source images can never fall in more than one view at a time when multiple views are arrayed about a single station point. Therefore, it is practical to compute a view assignment for the points and a common computing unit may serve a number of views.

B. SYSTEM CONFIGURATION

The block diagram of Figure 6 shows the principal elements of the image-generating system. It is divided into the following four subsystems:

1) General-purpose Computing
2) Surface Generation
3) Object Generation
4) Point Source Generation

The General-purpose Computing subsystem serves all three image-generating subsystems. It accepts inputs from the flight dynamics computer which specify the position and attitude of the aircraft. It performs computations related to the motion of the aircraft and supplies the basic data required by the other subsystems.

Two blocks, in Figure 6, are shown as being part of two subsystems because, functionally, they serve their respective subsystems; physically, however, they are part of the general-purpose computing equipment. The object environment memory stores the vertex, face, and color information for all three-dimensional objects. It also serves as buffer storage for priority information and the display vector quantities required by the Object Computing Section. The Point Source Computer is a separate processor that calculates the line and element numbers of the points.

Each of the image-generating subsystems operates independently. Each frame, new data are received from the general-purpose subsystem and video data are supplied in real-time to the Video Processors. Here, the data are combined, decoded, and converted into analog voltages to drive the displays.

The video data words supplied to the Video Processors consist of six-bit numbers. These numbers are used in one of two ways, depending upon whether the system is to be used for color or black and white. For black and white operation, the numbers are converted directly to analog voltages. For color operation, the number is used as an address to access a 64-word color memory. Each word in the color memory contains three numbers that specify the intensity of the red, green, and blue components of the desired color. The color memory and additional digital-to-analog converters are the only extra components involved in providing color.

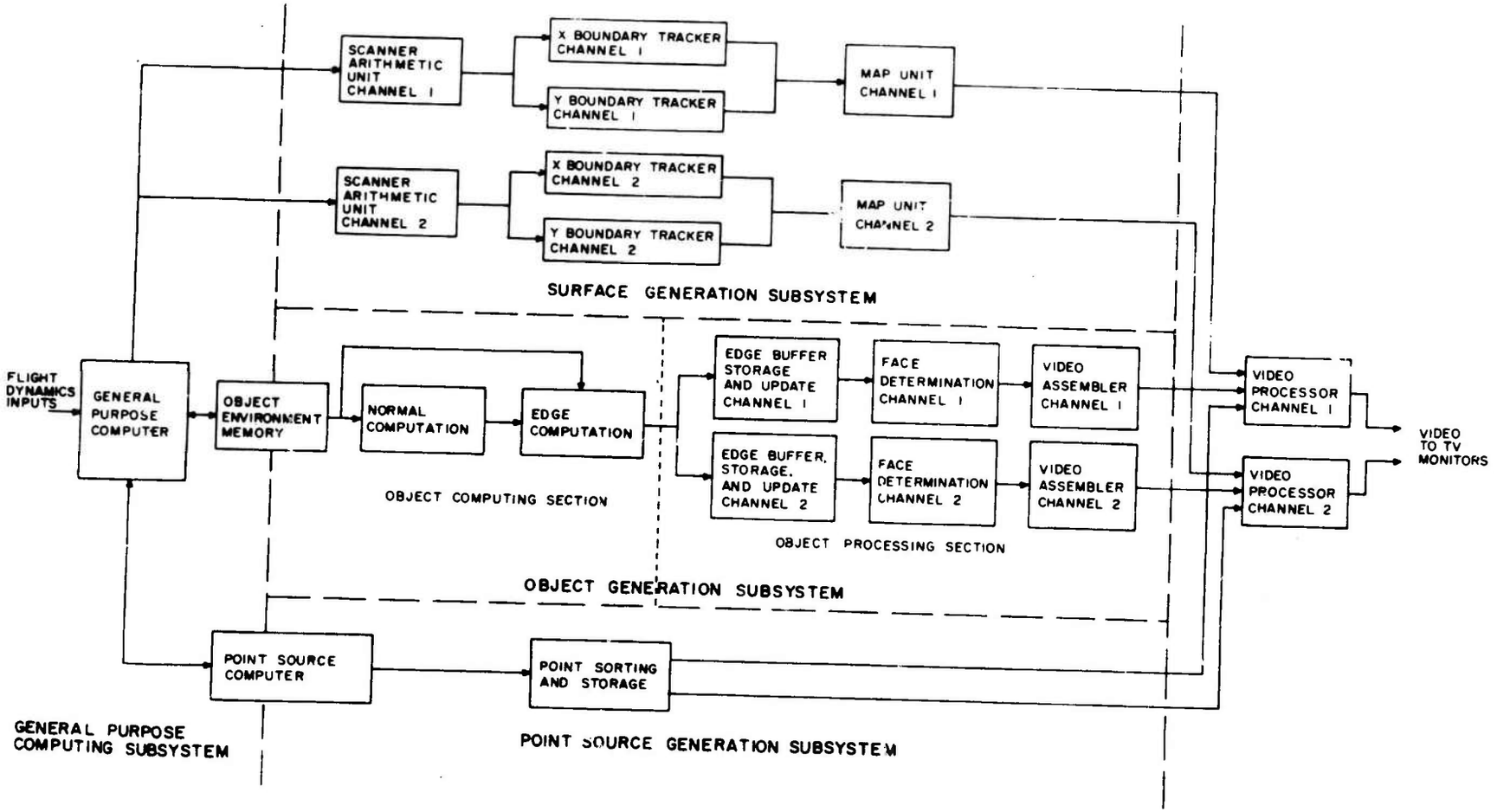


Figure 6. Image Generator Block Diagram

The Video Processors combine the video data generated by each subsystem using a fixed priority relationship. The surface generator output has the lowest priority, which means that its data are displayed only when no other data are present. Point-source video has the next highest priority and is displayed in preference to the ground surface. Object video is always displayed when present. This method of combining the video is only valid for point sources at ground level such as runway lights. The relative priority of point sources and objects would be reversed for representing tracer fire, neglecting the case where rounds drop behind objects (which occurs infrequently).

The Object Generating Subsystem is divided into two sections. The computing section performs the vector mathematics described previously. Its output is the set of A and B edge parameters. Both computing blocks have sufficient capacity to process 1000 edges and can therefore provide data for both channels. The object processing section constructs the display plane image from the edge parameters. Data are prepared line-at-a-time in the Video Assembler.

Each of the blocks shown in Figure 6 is discussed in more detail in subsequent sections of this report. Functional requirements, limiting design factors, estimated complexity, and possibilities of expanded capability are included in the discussions.

C. SYSTEM PARAMETERS

The conceptual system design is based on the parameters discussed below. The selected parameters were chosen on the basis of operational requirements and practical factors affecting the implementation. A short discussion is included in each case, to show the rationale behind the choice.

1. Field of View

The computed field of view for each display channel is controlled by a numerical constant. Although it is possible to scale the computations for any field of view between 0 and 180 degrees, it is not practical to accommodate the entire range within one program. In addition, the extremes of the range are not usually of practical value. Therefore, a working range of 45 to 120 degrees is assumed. This angle refers to either the display width or height and there is no necessity for the two to be equal.

2. Raster Parameters

The design of the image generator depends strongly upon the choice of the television line standard. On the assumption that vertical and horizontal resolution are to be comparable, the number of picture elements to be computed varies as the square of the number of display lines.

The line standard not only affects the computed video rate, but is especially significant in the design of the Object Processing Section (see Section VIII). In particular, the amount of hardware in the Video Assembler is directly related to the number of picture elements on a raster line. For this reason, it is impractical to provide programmable control over the line standard. The system is therefore based on a 1023-line display with the following characteristics:

Frame rate - 30 fps
Interlace - 2:1
Active lines and elements - 1000 nominal
Line period - 32.6 microseconds
Active line time - 27 microseconds
Video clock rate - 40 MHz

3. Coordinate Systems

The most complex training task requires two dynamic coordinate systems (one for each aircraft) in addition to the ground, or fixed, reference system. One additional coordinate system is provided for each view. This allows the station point to be moved relative to the center of gravity of the aircraft and the sight line to be offset from the aircraft's forward axis.

4. Range and Range Resolution

These quantities refer to magnitude and precision of the X, Y and Z displacements of the dynamic coordinate systems, relative to the reference system origin. There are no inherent limitations on these quantities, but the following values can be accommodated with single precision computations:

Translational Range - $\pm 2^{20}$ feet (approx. 175 nmi.)
Altitude - 0 to 2^{17} feet (approx. 20 nmi.)
Resolution (least increment) - 2^{-5} feet (translation and altitude)

5. Input Interface

The image generator requires flight dynamics inputs that specify the position and attitude of the aircraft and certain control information. A complete set of inputs is required once per frame. An all-digital interface is assumed.

The position of each dynamic coordinate system is specified by the X, Y and Z coordinates of its displacement from the reference origin. These input quantities may be either whole numbers or incremental. In the former case, the complete position description is supplied each frame, requiring a 32-bit word for each quantity. In the latter case, the problem is started from some initial point and the dynamics computer supplies the changes in position from frame to frame. The choice of format is unimportant at this time. A total of six data words is required to describe the positions of both aircraft.

The attitude inputs may also be provided in a number of different formats. One method of defining attitude is by means of Euler angles. If Euler angles are used, the inputs may be the three angles, the sines and cosines of the three angles, or related incremental quantities. Often, the dynamics computer has available the direction cosine matrix relating the aircraft coordinate system and the reference system. This is the most convenient format for the image generator. A total of 18 words would be required for both aircraft. These data should be supplied with 16-bit precision.

A small amount of control information is required. These bits would be used to initialize or hold the position of the dynamic coordinates systems, signal weapon firing, etc. A single data word suffices. The entire data communication between the dynamics computer and the image generator consists of a 25-word transfer each frame.

6. Output Interface

The output consists of analog video signals and synchronizing pulses for the television monitors. Separate synchronizing signals are provided and, in the case of color output, it is assumed that the monitors accept direct red, green and blue video drive. All signals would be provided on 92-ohm coaxial cable with a nominal amplitude of 3 volts peak-to-peak.

IV. PRIORITY

A. BACKGROUND

One of the most significant problems that must be faced in the real-time computation of images is the priority, or hidden-line, problem. In our everyday visual perception of our surroundings, it is a problem that nature solves with trivial ease; a point of an opaque object obscures all other points that lie along the same line of sight and are more distant. In the computer, the task is formidable. The computations required to resolve priority in the general case grow exponentially with the complexity of the environment, and soon they surpass the computing load associated with finding the perspective images of the objects. This occurs because the problem is a relational one where, at least implicitly, a point on an object must be compared with other points on that object as well as with every other object, in order to reach a decision as to its visibility. Fortunately, many simplifications result when the objects are restricted to non-interesting polyhedra with convex faces. In this case, the problem is reduced to one that grows as the square of the number of convex objects. This, too, soon becomes intolerable. It means that advances in computing technology (or increased expenditures) would not lead to proportionate improvements in image complexity. The solution of the priority problem would always be limiting.

The hidden-line problem has received much attention recently from workers in the computer-graphics field. Most of these efforts have dealt with the problem as it applies to line or vector drawing displays. The difficulty is even greater here, because the picture information is presented sequentially, object line by object line. Thus, if a portion of a line is hidden, this fact must be computed and the line segment appropriately modified before it is displayed. In addition, since the display refresh time is fixed by other considerations, the complexity of the image is limited by the display writing speed. Both of these problems are eliminated in a raster scan system. Parallel computing channels may be employed to accommodate increased scene complexity and, with object data available in parallel, priority may be implemented by a technique analogous to video insetting.

Nevertheless, relatively few people have been concerned with computing images for raster scanned displays, and this work has primarily centered around the use of large general-purpose computers in non-real-time. Warnock[2], at the University of Utah, has achieved significant results with his hidden-line algorithm, which can handle intersecting polyhedra. Images containing in the neighborhood of 2000 polygons required approximately four seconds of computing time on a Univac 1108. The algorithm seeks out complex areas on the image plane and resolves these while skipping over portions that contain no new information. The computing time is a function of the average image complexity, but the time spent in computing a particular area of the picture may be relatively long. This concept of "frame averaging" is not readily

extendable to a real-time system organization because the computing power and the data buffering requirements in several segments of the system are strongly dependent on certain image-plane statistics that are not well known at this time.

Real-time solution of the priority problem was implemented in the NASA II system by a priority matrix technique which is described in the next section. It is a direct solution that may be used (at least in concept) to handle any environment that can be constructed from non-intersecting polygons. It has two drawbacks. The hardware complexity increases with the square of the number of objects to be generated; and associated computations, although somewhat dependent upon the geometric arrangement of the environment, also tend to increase as the square of the number of objects.

The conviction that future systems must avoid the type of limitations inherent in the matrix approach led to the development of what is termed a "priority list" technique and a system organization based on it. The priority list technique reduces the problem to one that is linearly dependent on the complexity of the environment. This feature is obtained by sacrificing some of the previous freedom to arrange objects geometrically to form environments. This section of the report is concerned with the rigorous determination of the required geometric constraints - an initial step in the practical application of the concept.

B. MATRIX TECHNIQUE

The object-generating techniques described mathematically in Section III. A. deal basically with finding the perspective image of a polygon. The polygons (object faces) thus formed are necessarily convex. They are determined by the intersection of a set of half-spaces corresponding to the (infinite) edges in the display plane.

A convex body may also be described by its vertices. For example, the quadrilateral in Figure 7 has vertices A, B, C and D. They define the

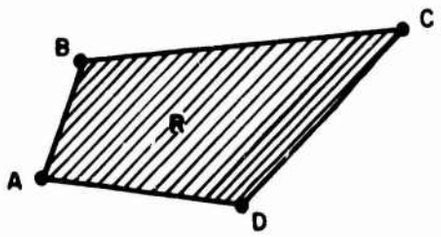


Figure 7. Convex Quadrilateral

convex region, R, which consists of all points that satisfy:

$$R = \alpha_1 A + \alpha_2 B + \alpha_3 C + \alpha_4 D \qquad (1)$$

for all α_i that satisfy:

$$\sum_i \alpha_i = 1, \quad \alpha_i \geq 0$$

This description is also valid for solids, in which case the vertices would not be coplanar.

It can be shown that any two non-intersecting convex bodies may be separated by a plane. A separating plane may always be constructed by finding a minimum distance line segment connecting the convex regions and placing the plane so that it is perpendicular to this line and contains a point in it. Figure 8 illustrates some conventions used to describe such planes.

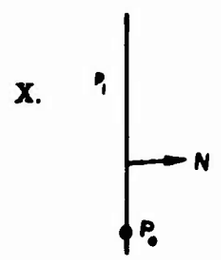


Figure 8. Separating Plane

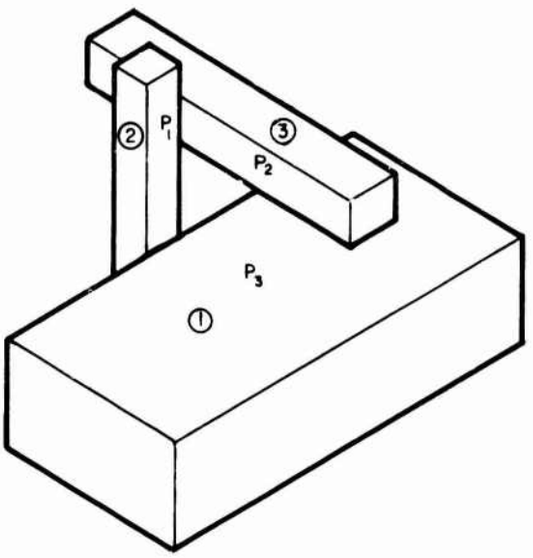
The name, p_1, is on the front (true) side, and the normal vector points out of the back (false) side. The true side consists of all points X for which:

$$X \cdot N \leq P_o \cdot N \qquad (2)$$

An observer, by determining which side of a separating plane he is on, can resolve any priority conflict that may arise. That is, if two faces are generated for the same display plane element, the correct one may be chosen. It should be noted that this priority determination depends only on the observer's position in the environment and not on his direction of view. Only one test need be made even if multiple views are to be computed from the same station point.

By using a few simple rules, the priority question may often be resolved among a group of faces without resorting to separating planes. An example is a convex solid. Actually, a thin-walled shell, rather than a solid, is generated. The outside faces of the solid would be given priority over the inside faces. If it were unnecessary to go inside the object, then the back faces would not be considered. No conflict can arise among the outside faces because the convexity condition assures that their display plane images do not have common areas. The term "object" will be used to indicate a priority entity which may consist of a single convex face, a convex object, or the convex hull of a group of objects.

A matrix is a convenient way to represent the pairwise priority relationships. The priority matrix, A, consists of the set of elements a_{ij} where $a_{ij} = 1$, if object i may obscure object j or $a_{ij} = 0$ if object i cannot obscure object j. Figure 9 shows a perspective drawing of three objects. In this



i \ j	1	2	3
1	0	p_1	$\bar{p}_3$
2	$\bar{p}_1$	0	p_2
3	p_3	$\bar{p}_2$	0

Original Matrix

i \ j	1	2	3
1	0	1	0
2	0	0	1
3	1	0	0

Matrix Evaluated for Vantage Point Shown

Figure 9. Priority Situation and Matrix

case, as is almost always true, faces of the objects could be found to serve as separating planes. The original matrix shows the separating planes as entries where the bar over an entry denotes the logical complement (false side of plane). The right-hand matrix is evaluated for our viewing position. The value of the matrix entries is controlled by our position with respect to the separating planes.

The matrix entries must be evaluated as the observer's position changes (generally once per frame time). In hardware, the matrix may be thought of as a cross-bar switch whose connections are alterable at the frame rate. The video associated with a column object would be displayed unless a "one" were present in its column and the corresponding row object were being generated at the same time. In other words, if unblanked object video is applied to the rows and the "ones" denote connections to the columns, the column outputs would be blanking signals. These priority decisions are made on an element-by-element basis.

Some of the disadvantages of this approach for large numbers of objects are obvious; others are more subtle. The size of the matrix and its physical analog can become prohibitive. If the switching network approach is implemented, then all of the object data must be available in parallel and the circuits must respond at the video rate. Other implementations are possible that would allow serial priority comparisons. The important observation is that not only must the matrix data be stored, but all object data must be accessible for priority decisions to be made. Thus, an indeterminate amount* of object data (some of which may be blanked) must be retained until a decision can be reached.

C. PRIORITY LIST TECHNIQUE

Most of these drawbacks disappear if it is possible to work with a list of objects that are ordered by their priority ranking. If the list were ordered, for example, from highest to lowest priority, then the first object in the list could have no other objects take priority over it, and the last could not take priority over any of the others. One advantage to such an approach can be seen immediately. Once the first object is processed and its display plane image is known, then any data pertaining to subsequent objects that occupy the same display elements may be discarded at once, because such data would be hidden. The image may be built up in this way, object by object, retaining only the visible image data at each step. An analogous procedure could be used to process the list in reverse order, if desired.

There is, of course, no matrix to be stored when the priority list approach is used, because the priority information is contained in the processing order. If n objects are involved, the priority computation consists of finding a list of length n rather than an array of size n^2.

It can easily be seen that some priority situations that can be represented by a matrix cannot be processed in a list. For example, no list can be formed for the scene shown in Figure 9. With reference to the matrix, one finds that: (1) may obscure (2); (2) may obscure (3); and (3) may obscure (1) -- an unending chain. It is important to note, though, that this chain could be broken by constructing one of the objects by using two smaller ones (at some cost in the number of edges used). One possibility would be to cut object (3) into two parts with plane p_1. The part hidden by (2) would be called object (4). A list could then be found for this environment for all possible vantage points.

Other examples of what is called an "improper" environment can be conceived - some simple, some intricate. They are all characterized by a closely packed, interlocking structure. It appears that many useful environments are also "proper" ones in the priority sense. It is also likely that, if an environment designer could readily distinguish between a proper and improper situation, he could often avoid the latter without sacrificing his artistry. If all else fails, he can always solve the problem by splitting an object into two.

---

* The amount will depend on the "depth" and extent of priority conflict and is environment-dependent.

D. THE PROPER ENVIRONMENT

It is not too difficult to distinguish proper and improper environments when only a few objects are involved. The real problem arises with practical environments containing many objects. A set of rules is needed for constructing proper environments (or detecting improper ones) which places the weakest possible constraints on the environment. This set of necessary and sufficient conditions is derived by considering the restrictions that must be imposed on the priority matrix. The first step is to find the allowable patterns of ones and zeros in a matrix evaluated at a particular vantage point. The conditions on the general matrix are then established.

1. Graphical Analysis

Some of the concepts of graph theory are useful in analyzing the priority problem. The graph is also a convenient way to visualize certain relationships. A directed graph consists of a set of nodes and arcs. An arc connects two nodes and is directed from one node to another by an arrow. For the priority problem, the nodes represent objects and the arcs indicate the relation "may take priority over". A directed graph, G, may be associated with a priority matrix. G would contain a node for each object, say, v_1, v_2, ... v_n. An arc would be directed from v_i to v_j if the matrix entry $a_{ij} = 1$; otherwise, the arc is omitted. The graph shown in Figure 10 shows the priority relationships for the objects of Figure 9.

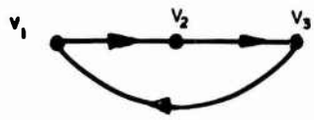


Figure 10. Graph Associated with Matrix of Fig. 9.

The arcs in this graph show all of the necessary relationships between the objects as specified by the matrix. The fact that there is an arc from v_1 to v_2 and another from v_2 to v_3 does not imply a relationship between v_1 and v_3. In considering a list structure, however, the implied relationships are important. If the list is to be valid, then the graph associated with it must not imply relationships that contradict the matrix (or its graph). Implied relationships are found by following paths that are directed sequences of arcs. If a path exists from v_i to v_j, then v_j is said to be reachable from v_i. A circuit is a path whose initial and terminal nodes coincide. The graph of Figure 10 contains a circuit.

We wish to form an ordered sequence of objects v_{i1}, v_{i2}, ..., v_{in}, such that v_{i1} has the lowest priority and v_{in} has the highest. It is not necessary that this list be unique; only that it not violate any of the relationships dictated by the matrix. The graph associated with the matrix must not contain any arcs from v_{ij} to v_{ik} for any $j < k$.

It is obvious that no list can be formed if the graph of the matrix contains a circuit. The list assigns a priority level L to each object. In our example of a circuit, we would require $L(V_1) < L(V_2) < L(V_3) < L(V_1)$, which is contradictory.

The converse can also be shown, i.e., if the graph does not contain any circuits, then a level assignment or list can always be formed. Let the original priority matrix be A and its associated graph, G. Assume that the objects in the matrix are numbered $v_1, v_2, \ldots, v_n$. If there are no circuits in the graph then there is at least one node with no outgoing arcs. (If no such node exists, then an infinite directed sequence of arcs could be formed and, since there are a finite number of nodes, the path would eventually close on itself at some node forming a circuit.) Label the node v_{i1}. If there are others, assign them labels $v_{i2}, \ldots, v_{ik}$. Form a new matrix A' where these are the first k objects and a new graph $G' = G - \{v_{i1}, v_{i2}, \ldots, v_{ik}\}$. Certainly G' does not certain any circuits. Therefore we may find at least one node of it which has no outgoing arcs. This process may be continued until all nodes are relabeled, and the matrix A' is completed. A' will be strictly lower triangular, because there are no arcs going from v_{ij} to v_{ik} for $j < k$ by construction. Therefore, the matrix associated with any graph that does not contain circuits can be reduced to triangular form by a permutation matrix Q:

$$A' = Q^t A Q \tag{3}$$

A priority level assignment may be made for each object of A' by letting $L(v_{ij}) = j$. The higher the level number, the higher the object's priority. The main diagonal entries in the matrix are always zero ($a_{ii} = 0$ for all i) since an object cannot take priority over itself. The matrix is asymmetric in the sense that $a_{ij} + a_{ji} \leq 1$ for all $i \neq j$. If all objects are separated by planes, then the equality holds and the matrix represents a complete order, that is, the level assignment is unique.

From the above, we may conclude that the following statements are equivalent:

1) The graph G associated with A contains no circuits.
2) A level assignment or list may be formed.
3) The matrix A may be reduced to strictly triangular form.

2. Matrix Criteria

A more analytical statement is needed for the matrix conditions that guarantee that there are no circuits in the graph. Alternate criteria for the matrix A may be found which are more useful in further analysis.

The square matrix A has elements a_{ij} which may take on the boolean values zero or one. Circuits of length one are precluded by stipulating that $a_{ii} = 0$ for all i. If there is a circuit of length two through node i, then there are two arcs with corresponding unit matrix entries at a_{ik} and a_{ki}. Since these are non-negative numbers, circuits of length two are precluded by requiring:

$$\sum_k a_{ik}\, a_{ki} = 0 \text{ , for all i} \qquad (4)$$

The summation of (4) is the i^{th} diagonal entry, $a_{ii}^{(2)}$ of the matrix A^2. The parenthetical superscript denotes that the element is associated with second power of the matrix. Circuits of length two do not appear if A is asymmetric ($a_{ij} + a_{ji} \leq 1$). In general, the (i, j) entry of A^r is the number of paths of length r between nodes i and j. For a matrix of order n, we need be concerned only with circuits of length n or less, since any longer ones must necessarily include shorter ones. Therefore, if all of the diagonal elements of the matrices A, $A^2, \ldots, A^n$ are zero, then the associated graph contains no circuits. This may be expressed mathematically in terms of the trace of the sum of the matrices:

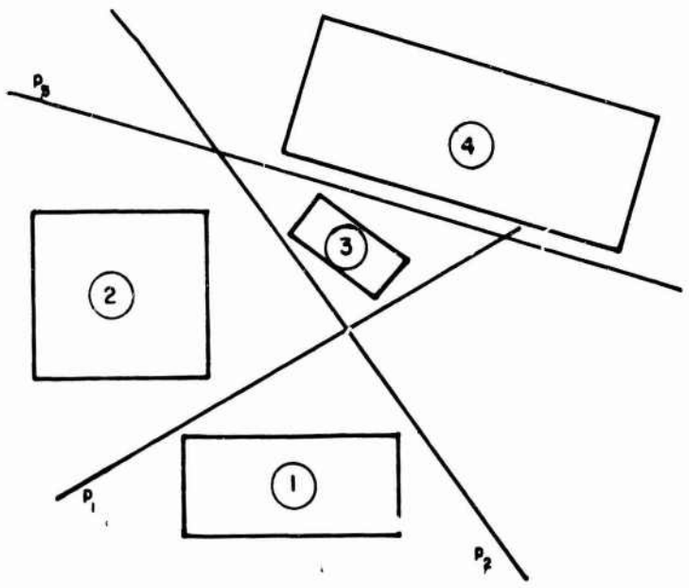
$$\text{tr}\left[A + A^2 + \ldots + A^n\right] \quad \begin{cases} = 0 \longrightarrow \text{priority list exists} \\ \neq 0 \longrightarrow \text{no priority list exists} \end{cases} \qquad (5)$$

Since it is only necessary to know whether this expression is zero (or not), it can be simplified by replacing the ordinary arithmetic operations of multiplication and addition by boolean operations. Multiplication and addition are carried out in the usual manner, except that $(1 + 1)_B = 1$ and $(k)_B = 1$ for any non-zero integer k. The subscript B indicates a boolean operation. Boolean operations are assumed in all of the following and subscripts will be omitted. It follows that the (i, j) element of A^r is a one if there are any paths of length r between i and j and it is zero otherwise.

Up to this point, we have been considering matrices with numerical entries such as would be obtained for a particular position in an environment. In the general case, the entries are logical rather than numerical and consist of the separating plane variables. We are interested in knowing that a list may be found not only for a particular position in the environment, but for all possible positions. However, it is generally not necessary to consider all of the possible matrices that could be obtained by assuming that the separating plane variables can take on independent states. If there were s separating planes, 2^s different matrices might result. Usually, the geometry of the separating planes precludes some of these states. Figure 11 shows a plan view of a simple environment, where the separating planes and the objects extend vertically out of the page. In this example, the state $(p_1 p_2 p_3)$ cannot be obtained, since it is impossible to be on the true side of all three planes simultaneously.

What conditions are necessary to assure that the diagonal terms of the first n powers of A are zero, as required by (5)? The matrix entries, a_{ij}, are thought of now as separating plane variables such as p_k and $\bar{p}_k$, or they may be identically zero. We must have:

$$a_{ii}^{(r)} = 0 \qquad \text{for all indices, i, and powers, r, from 1 to n.} \qquad (6)$$



	1	2	3	4
1	O	p_1	p_2	$\bar{p}_3$
2	$\bar{p}_1$	O	p_2	p_2
3	$\bar{p}_2$	$\bar{p}_2$	O	$\bar{p}_3$
4	p_3	$\bar{p}_2$	p_3	O

Matrix (1)

	1	2	3	4
1	O	p_1	p_2	$\bar{p}_3$
2	$\bar{p}_1$	O	p_2	$\bar{p}_3$
3	$\bar{p}_2$	$\bar{p}_2$	O	$\bar{p}_3$
4	p_3	p_3	p_3	O

Matrix (2)

Figure 11. Priority Example

For r = 1:

The condition is always satisfied, because the main diagonal elements of A are identically zero.

For r = 2:

The condition is satisfied by the asymmetry of A. Denoting the i^{th} row vector by a_i and i^{th} column vector by a^i, we have:

$$a_{ii}^{(2)} = \sum_k a_{ik} a_{ki} = a_i \cdot a^i = 0 \qquad (7)$$

since either $a_{ik} = \bar{a}_{ki}$ or $a_{ik} = 0$

For r = 3:

It can be shown that:

$$a_{ii}^{(3)} = a_i^{(2)} \cdot a^i = a_i A a^i \tag{8}$$

It is helpful to write this expression out in the following form and view it as a dot product of the vector a_i with the column vector (Aa^i):

$$a_{ii}^{(3)} = (a_{i1}, a_{i2}, \ldots, a_{in}) \begin{bmatrix} A \end{bmatrix} \begin{pmatrix} a_{1i} \\ a_{2i} \\ \cdot \\ \cdot \\ a_{ni} \end{pmatrix} \tag{9}$$

For $a_{ii}^{(3)}$ to be zero when some $a_{ij} = 1$, we need the j^{th} element of the column vector (Aa^i) to be zero. If some $a_{ij} = 0$, then the j^{th} element of (Aa^i) may be either one or zero. Therefore, we require:

$$a_j \cdot a^i \leq \bar{a}_{ij} \text{ for all i and j} \tag{10}$$

$$\text{or in matrix form: } Aa^i \leq (\bar{a}_i)^t$$

The inequality applies to each component of the vector.

If each object is separated from every other object by a plane, then the matrix is complete and $a_{ji} = \bar{a}_{ij}$ for all $i \neq j$. This is the case of practical interest and allows the substitution of a^i for $(\bar{a}_i)^t$. The conditions of (10) then become:

$$a_j \cdot a^i \leq a_{ji} \text{ for all i and j} \tag{11}$$

$$\text{or in matrix form: } Aa^i \leq a^i \text{ for all i} \tag{12}$$

This relationship also assures that the diagonal terms of all higher-order matrices will be zero, since:

$$a_{ii}^{(r)} = a_i A^{r-1} a^i \leq a_i \cdot a^i = 0 \tag{13}$$

Therefore, the necessary and sufficient conditions for the complete matrix A to represent a proper environment are given by (11). These results may be interpreted graphically by considering the expression for a path of length r between nodes i and j:

$$a_{ij}^{(r)} = a_i A a^j \leq a_i \cdot a^j \leq a_{ij} \tag{14}$$

If such a path exists, then there must be a direct arc from i to j. Certainly, there is no arc from j to i if there is one from i to j (because there is a separating plane between i and j). Hence, there is no possibility of a circuit.

The application of (11) to detect an improper situation can be illustrated by considering matrix (1) in Figure 11. The matrix is a valid representation of priority for the environment shown. Consider the term a_{42}. According to (11), we must have:

$$\bar{p}_2 \geq a_4 \cdot a^2 = p_3 p_1 + \bar{p}_2 \cdot 0 + \bar{p}_3 p_2 + 0 \cdot \bar{p}_2 \tag{15}$$

The second and fourth terms are obviously zero. The third term is no problem, since it contains $\bar{p}_2$. Multiplying through by p_2, we find that $p_1 p_2 p_3$ must always be zero. This is not generally true; however, in this case, the geometry prohibits this condition, as mentioned previously. A similar examination of the condition for a_{12} requires $\bar{p}_1 \bar{p}_2 \bar{p}_3$ to always be zero. This is not the case and, in fact, no list can be formed from this matrix when the observer is on the false side of all three planes.

Matrix (2) represents the same environment and the same set of separating planes. A check of its terms shows that a list can always be formed. It is therefore possible to have a proper environment, but an improper choice (or use) of separating planes will make it impossible to obtain a list in certain cases.

3. Selection of Separating Planes

It would be much more useful to the environment designer to have an interpretation of (11) that would allow him to synthesize environments and select separating planes, rather than one that merely pointed out his errors. To this end, we re-examine (11).

Multiplying both sides of (11) by a_{ij} and recognizing that either $a_{ij} = \bar{a}_{ji}$ or $a_{ij} = 0$ yields:

$$a_{ij} (a_j \cdot a^i) \leq a_{ij} a_{ji} = 0 \tag{16}$$

for all i and j. Expanding the row and column vector, the requirement becomes:

$$a_{ij} a_{jk} a_{ki} = 0 \tag{17}$$

for all i, j, and k. This equation may be satisfied by one or more of the following conditions:

1) At least one term is a diagonal element (identically zero).

2) Two of the terms are complementary plane variables (e.g., if $a_{ij} = p_s$ and $a_{ki} = \bar{p}_s$).

3) The geometry of the situation is such that all three plane variables cannot be simultaneously true.

The environment designer need not be concerned with condition (1). Conditions (2) and (3) only apply when i, j and k are all different, since the other cases are included in (1). Therefore, we are concerned with cases involving three distinct objects and their separating planes.

If it were not for the possible cases admitted by (3), then it could be concluded that (2) must always be satisfied. Condition (2) is satisfied only by having two of the three objects separated from the third by the same plane as shown in Figure 12. Here, i is separated from j and k by p_s, while j and k are separated by a second plane. Thus, three objects must be separated by two planes.

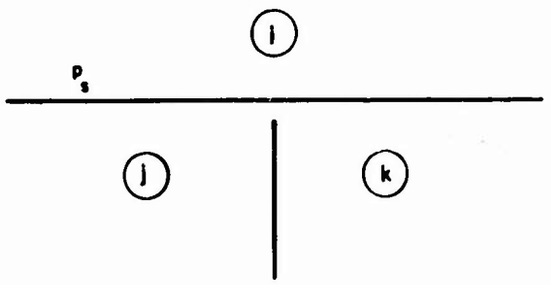


Figure 12. Proper Separating Planes

What other cases are possible under condition (3)? Each of the three terms of (17) are distinct planes in this case and each may be represented by an inequality of the form given in (2). The three inequalities may be combined and written in matrix form as:

$$[C]\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \leq \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix} \tag{18}$$

where C is a matrix whose row vectors are related to the normals of the separating planes, and k_1, k_2, and k_3 are associated with the distance of the planes from the origin. The set of simultaneous inequalities will have no solution, as required by (3), only if the rank of C is less than three. This occurs only when the three normals can be contained in a plane.

If the rank of C is one, then the three planes must be parallel as shown in Figure 13. No matter how the objects are arranged with these as separating planes, one of the planes is superfluous and the problem can be recast using two planes as for condition (2).

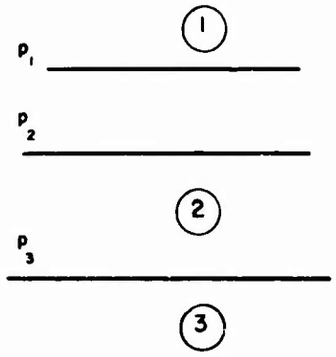


Figure 13. Case where Rank of C = 1

If the rank of C is two, then the three planes must be perpendicular to a common plane. One of these configurations is shown in Figure 14, where two of the planes are parallel. The space is divided into six regions,

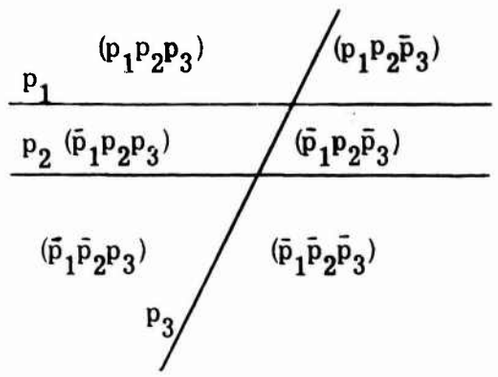


Figure 14. Configuration (A) where Rank of C = 2

which implies that two of the eight possible states can not be reached. They are $(p_1\bar{p}_2p_3)$ and $(p_1\bar{p}_2\bar{p}_3)$. These are states that the geometry will not allow. If one of these states arises from considering (11) for element a_{ij}, then a corresponding state will also arise where each term is complemented when element a_{ji} is considered. Since both complementary states $(\bar{p}_1p_2\bar{p}_3)$ and $(\bar{p}_1p_2p_3)$ do exist, there is no way that this configuration could be used to properly separate three objects. It can also be shown, by enumerating the possibilities, that any arrangement of objects that can be separated by planes of configuration (A) can also be separated by only two of them and, hence, handled by condition (2).

Configuration (B) shown in Figure 15 is a second case that arises when the rank of C is two. Here, the three normals are in a plane, but no two are colinear. Seven of the eight possible states exist. By the previous argument, the one non-existant state is of no help since its complement does

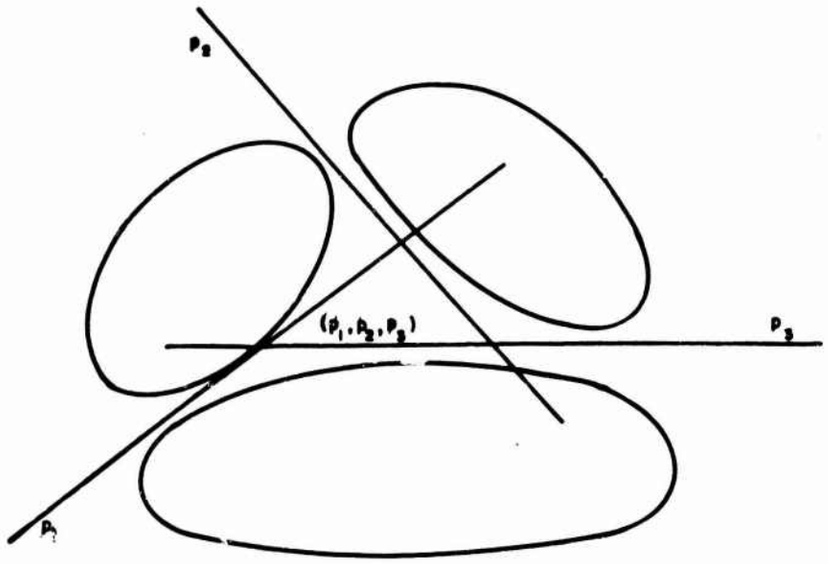


Figure 15. Configuration (B) where Rank of C = 2

exist and would be encountered in the symmetric matrix term. It should also be noted that there is only one arrangement of objects, within this separating configuration, that is prohibited and that is the one indicated in the figure. All others may be separated by two of the three planes.

The final case is a degenerate version of the last one, and is shown in Figure 16. The three planes intersect in a line. Here, the two states

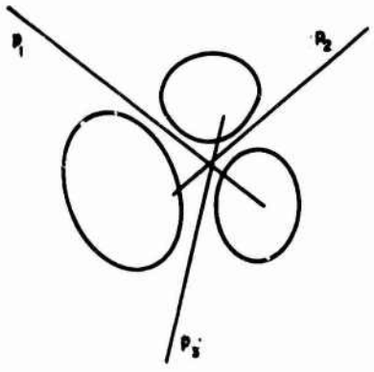


Figure 16. Configuration (C) where Rank of C = 2

that are excluded are $(p_1 p_2 p_3)$ and $(\bar{p}_1 \bar{p}_2 \bar{p}_3)$, which are complements. For the arrangement of objects shown, these are the only two terms that would arise in considering the matrix terms, and the case is allowed. Because the definition of the true side of the plane in (2) includes the equality, the line of intersection would actually yield the condition $(p_1 p_2 p_3)$. The case can still be accommodated if it is recognized that the order of priority is unimportant for this state and it is handled accordingly. This configuration is termed the "star" case.

E. CONCLUSIONS

The geometric restrictions that must be observed in constructing a proper environment and set of separating planes suitable for priority list processing have been derived and may be summarized as follows. Every subset of three objects in the environment must either be 1) separated by two planes, or 2) a star configuration.

This means that the number of separating planes and the number of tests which must therefore be made increases only linearly with the number of objects. The matrix, although useful in considering the restrictions, need not be formed to obtain the list.

The imposed restrictions can be readily visualized and do not appear to be particularly limiting for the types of environments needed for pilot training. It should be noted that both the matrix and the list technique apply mainly to environments or portions of environments that are fixed in space where the separating planes are determined off-line. Moving objects such as would be encountered in formation flying or air-to-air combat can be accommodated provided that they are easily separated from the remainder of the environment (by operating above the terrain, for example).

Further investigation is needed to (1) detail the list-forming algorithm, (2) provide computer-assisted environment design, and (3) find methods of working with more complex moving object situations.

V. STATUS OF THE SEMICONDUCTOR INDUSTRY IN DIGITAL INTEGRATED CIRCUITS

A. INFLUENCE OF CIRCUIT TECHNOLOGY ON SYSTEM DESIGN

The first consideration in the system design process is, "What kind of equipment is available to work with?" Several years ago, the most important part of the answer to this question might have consisted of a list of available transistors, along with their frequency capabilities, breakdown voltages, and so on. More recently the answer has become complicated by the broad availability of families of integrated circuits, necessitating judgements based on power consumption, logic speed, noise characteristics, and many other parameters. However, the present and future availability of large and medium-scale arrays adds further dimensions to the problem of defining a set of ground rules for the design of a large system. Important considerations are:

1) What kinds of circuits and arrays are available now?

   In the two broad areas of technology - bipolar and MOS circuits - what kinds of advances are to be expected, both in the future and over the long term?

2) Are custom programs available?

   How expensive are the various approaches now? How expensive are they going to be in the future?

3) What manufacturers are involved in the various technologies? How likely are they to be able to make what they say they can make?

The answers to these questions and to several others all affect the initial system design decisions - how the computation will be broken down and how each part of the computation will be carried out.

The results of literature searches and trips to semiconductor vendors and the way in which these results affect the design of various parts of the system are described in this section.

B. DEFINITIONS OF TERMS

"Large-scale integration" and "medium-scale integration" are probably the terms that occur most frequently in semiconductor advertising today. No precise definitions for these terms exist. Sometimes it appears that MSI is the term one manufacturer uses to describe another manufacturer's more complex products, whereas he calls his own LSI. Rather than attempt to generate yet another definition, or to subscribe to an arbitrary categorization on the basis of number of gates or number of layers of metallization, the terms LSI and MSI will be used, in this section, to refer to the following broad categories, which may overlap:

MSI Array - a circuit containing several logic gates or flip flops interconnected to perform a simple logical function. This category excludes such things as quad latches and hex inverters, but includes short shift registers, counters of a few stages, and decoders.

LSI Array - a circuit containing many logic gates or flip flops interconnected to perform a complex logic function. Examples include multiple shift registers of 16 or more bits, 8-or-more-bit parallel adders with carry look-ahead, and up-down simultaneous counters with eight or more stages.

The terms "standard" and "custom", as applied to integrated circuits, also need explanation because between the group of standard, distributor stock circuits and the group of custom designed circuits involving tailored layout, mask sets, and so on, there exists a group of what might be called standard custom circuits. These circuits involve the use of standard semiconductor chips containing several cells, where a cell may be a gate, a flip flop, or a simple logic function. The make-up of a standard chip is fixed, and the chips are mass-produced and stocked. The interconnection of the cells on a chip is accomplished with a custom-designed mask, or set of masks, to produce the function the customer desires. Thus, the basic units (cells) of such an array are standard circuits, but the interconnection of these cells to form a function is accomplished on a custom basis. Such a standard array with custom interconnections will be referred to here as a variable array.

C. INDUSTRY STATUS IN MAJOR AREAS OF INTEREST

This section covers the findings of the literature search and vendor visits described above as they relate to the types of circuits that would be useful in a contemplated large system.

1. MOS Circuits

Metal-oxide-semiconductor (MOS) circuits have probably received more publicity than any other kind of medium or large-scale array, and in many cases the excitement is justified.* MOS circuits offer considerable advantages over bipolar circuits, both in cost and in the degree of circuit complexity that can be incorporated on a given chip area. Operating speeds of readily available MOS circuits are more than an order of magnitude lower than those of standard bipolar devices. However, this disadvantage does not preclude the use of MOS in systems having a large computing load. The real criterion for circuit suitability is computing power per dollar, and the cost advantage of MOS circuits may make it worthwhile to organize such a system around slow, register-like structures in order to gain this advantage.

Shift registers and memory circuits dominate the group of currently available MOS circuits, partly because registers and memories are complex circuits that have very general application, but primarily because their

* The appendix contains a description of the basic operation and processing techniques for MOS circuits, together with a discussion of advances in the technology that are anticipated in the near future.

metallized interconnections are very simple. Processing is therefore simplified and the chip area required for interconnections is minimized. Both of these factors emphasize the strongest advantages of MOS circuitry: simple processing and high component density.

The most common MOS circuits available are single or multiple shift registers of considerable length, with shifting rates of one or two MHz. Table II gives some examples of available registers.

TABLE II

MOS SHIFT REGISTERS

	No. of Registers Per Pkg.	No. of Bits Per Reg.	Clock Phases	Max. Shifting Rate, MHz
Fairchild	2	256	4	2
Texas Instruments	2	100	2	1
General Instruments	2	25	2	0.5
National Semiconductor	2	50	2	1

This summer Fairchild expects to introduce a multiplexed register, the 3324, illustrated in Figure 17. The circuit is composed of two 256-bit registers, each capable of shifting at 2 MHz, together with internal multiplexing and demultiplexing circuitry. Externally the unit appears to be a single 512-bit register capable of shifting at 4 MHz.

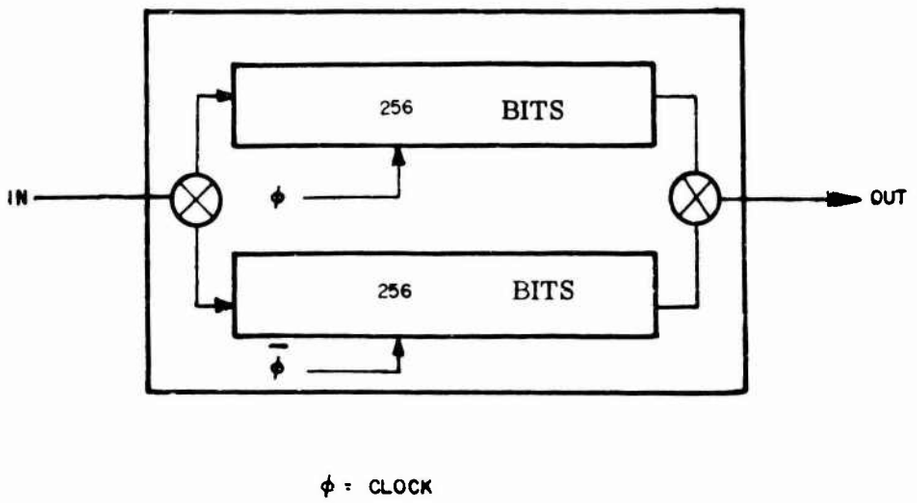


Figure 17. Fairchild 3324 Multiplexed Register

Memory circuits are available in both random-access and read-only configurations. One of the more complex read-only memories available is the Fairchild 3502, which contains 4096 bits and has an access time of about 800 nsec. This memory, organized into nine-bit words which are read out in parallel, is aimed at the character generator market. Fairchild also has two types of random-access memories available: the 3531 and the 3511. Both are 256-bit memories. The first has internal addressing logic and an access time of 1 μsec. The second consists of the memory array only, and can be externally addressed by fast logic through converters. Its access time is 300 nsec. Texas Instruments is nearly ready to introduce a similar 256-bit random-access memory array, which has a 100-nsec access time.

Both Texas Instruments and Fairchild have operational variable array programs utilizing MOS circuits. Because both are fairly new, it is difficult to make an accurate judgement of how effective these programs are. However, these programs do provide an excellent means for complementing the line of standard MOS products. Both involve computer-aided design for logic simulation (including the effects of gate delays, device parameters, run lengths, and processing variations). This simulation is intended to replace the conventional breadboard, which would not be able to approximate the finished circuit closely enough to indicate all design problems. These programs also employ automatic mask generation, utilizing a mask-cutting plotter driven by a computer, to generate the final interconnection mask which adapts to array to the customer's function. Finally, both employ computer-aided techniques to generate automatic or semi-automatic testing procedures for verifying the proper functioning of the completed circuits.

These variable array programs are capable of producing circuits involving 150 gates. Table III compares the cost and delivery time

TABLE III

COMPARISON OF CUSTOM AND VARIABLE ARRAY APPROACHES

Approach	Engineering Cost	Delivery Time	No. of Units for Economic Feasibility
Custom circuit	$50,000	9-12 months	$\geq$ 10,000
Variable array	$10,000	6-8 weeks	$<$ 10,000

for specialized circuits constructed by a variable array approach and by an entirely custom approach (i.e., an approach wherein the entire desired circuit would be designed to the customer's needs, rather than starting with an adaptable array). The variable array approach is much more attractive when moderate quantities are required. The all-custom approach, however, naturally makes more efficient use of silicon area, so that the cost per circuit will always be lower. If more than 10,000 units of a type are required, the per-circuit savings is sufficient to make the custom approach cheaper.

The real attraction of the variable array approach is that it allows a customer to have at an acceptable price in time and money, MOS circuits which perform specialized logic functions, which a manufacturer could never produce as standard products because of the specialized nature of the circuits. This approach, or some other customizing approach, is essential if large systems are to be built primarily of MOS circuits.

A number of advances in MOS technology, most of which are to be expected within the next year, are described in the appendix. These are aimed at increasing operating speed and at making the logic levels and driving capability of MOS circuits compatible with bipolar circuits. The success of these efforts will facilitate the utilization of MOS circuits by eliminating the need to design around all speed bottlenecks in a system. A designer could rely on high-speed bipolar circuits in these situations without the necessity for level conversion between MOS and bipolar devices.

Within the next year the proven feasibility of MOS variable arrays, the ready availability of 4 to 5 MHz circuits, and the production of circuits with bipolar-compatible logic levels could make the design of a large system almost entirely out of MOS components an attractive alternative.

2. Standard Bipolar MSI and High-speed Circuits

Several manufacturers have MSI circuits available as standard products. Representative types of these circuits, which generally are a part of the manufacturer's medium or high-speed logic lines, are given in Table IV. These circuits are either available now or are scheduled for introduction during the third quarter of 1969.

TABLE IV

EXAMPLES OF AVAILABLE MSI CIRCUITS

Circuit	Manufacturer	Package	Comments
1 of 16 decoder	Fairchild	24 pin	TTL levels
8 input multiplexer	Fairchild	16 pin	t_d 35 nsec
128 bit RAM	Fairchild	36 pin	50 nsec access
16 bit RAM	Motorola	14 pin	50 nsec access
16 bit data selector	T.I.	24 pin	TTL circuitry
8 bit parity generator	T.I.	24 pin	TTL circuitry

There are several hundred such different MSI circuits available throughout the industry, all of a generalized rather than a specialized nature. This is quite naturally the case. Manufacturers are not interested in making a circuit that one customer will buy, in small quantities. There is therefore, a continuing need for implementing parts of a system with standard integrated circuits or with some type of variable array. Two variable array programs for bipolar circuits are described below in Paragraph 3.

Several high-speed logic families are now on the market or are about to be introduced. Table V lists several characteristics of representative families.

TABLE V

HIGH-SPEED LOGIC CHARACTERISTICS

Logic Family	Mfr.	Speed Characteristics	Comments
MECL II	Motorola	4 nsec gates, 85 MHz FF	On mkt. for 18 mos.
MECL III	Motorola	1 nsec gate, 350 MHz FF	Question availability
9500 CML	Fairchild	2 nsec gates	1969 introduction
9800 CTL	Fairchild	4 nsec gates	1969 introduction
ECL 2500	T.I.	4 nsec gates	1969 introduction
TTL	Many	6 nsec gates, 25 MHz FF	Well established

Three years ago, industry watchers were predicting the early demise of emitter-coupled logic (then represented almost exclusively by Motorola's MECL family) and they indicated that TTL as the family that would make large high-speed computers practical. There were several reasons for this prediction. Logic designers were familiar with saturating logic circuits. TTL is saturating logic and offers many advantages over the earlier DTL and RTL families - primarily higher speed and push-pull output drive. MECL, on the other hand, is not saturating logic. Its logic swing is only 0.8 V peak to peak; neither logic level is at ground; noise immunity is only 150 mV on each logic level.

The objections to MECL turned out to be largely prejudicial, however. The difficulty of working with low logic levels and low noise immunity was mitigated by the fact that the logic type generates little noise. The major reason for the acceptance of emitter-coupled logic, however, was speed. TTL is certainly the fastest available saturating logic. Non-saturating emitter-coupled logic, however, has a built-in speed advantage. Consequently, both Texas Instruments and Fairchild are about to introduce ECL families competitive with Motorola's.

Motorola's MECL II family is the best currently available high-speed logic family. The family has been on the market for 18 months and it demonstrates Motorola's expertise in ECL technology. The selection of circuits is extensive, and many of the characteristics of MECL that originally made it somewhat difficult to apply have been eliminated. For example, MECL II circuits require only one power supply, whereas with MECL I it was necessary to use one MC 304 or MC 354 bias driver to generate a reference voltage for each 10 to 25 gates. High-speed clock drivers and double-ended line driver and receiver circuits, previously unavailable, are included in the MECL II line.

Fairchild and Texas Instruments have patterned their ECL logic families after Motorola's in many respects. The Texas Instruments line unfortunalely requires a separate bias driver, as did MECL. Fairchild's ECL family is considerably faster than MECL II and has one very significant circuit advantage. The Fairchild logic circuits incorporate a temperature compensating network that holds the logic levels within much narrower limits than is the case with MECL II. As a measure of how far advanced the development of the Fairchild and TI ECL families are, it is interesting to note that as yet neither family has a flip flop characterized.

Ironically, Motorola's own efforts at introducing an even faster ECL family have met with limited success. More than a year ago, the initial circuit characterizations for a MECL III family were revealed. These circuits were to have propagation delays less than one nsec; 400 MHz flip flops were expected. Not long after the initial characterization, three circuit types were introduced. These were the MC1660 dual four input gate, the MC1662 quad two input gate, and the MC1670 type D flip flop. Propagation delays for the gates were 1.1 nsec; that for the flip flop was 1.8 nsec. The maximum toggling frequency for the flip flop was greater than 300 MHz. Two of each type of these circuits could be purchased in an embossed plastic attache case for $150. Prices have since come down to $10 per gate and $18 per flip flop for quantities of 100, but these three circuits are still the only ones available. Rumors have it that Motorola's MECL III is having difficulty in making the jump from pilot line to full production. There is a warning here for the system designer who would base a projected system on circuits that are "almost ready to be introduced".

Another type of high-speed logic that is being prepared for market introduction is Fairchild's CTL II. CTL stands for Complementary Transistor Logic. The basic gate circuit is shown in Figure 18. CTL II is a high-speed

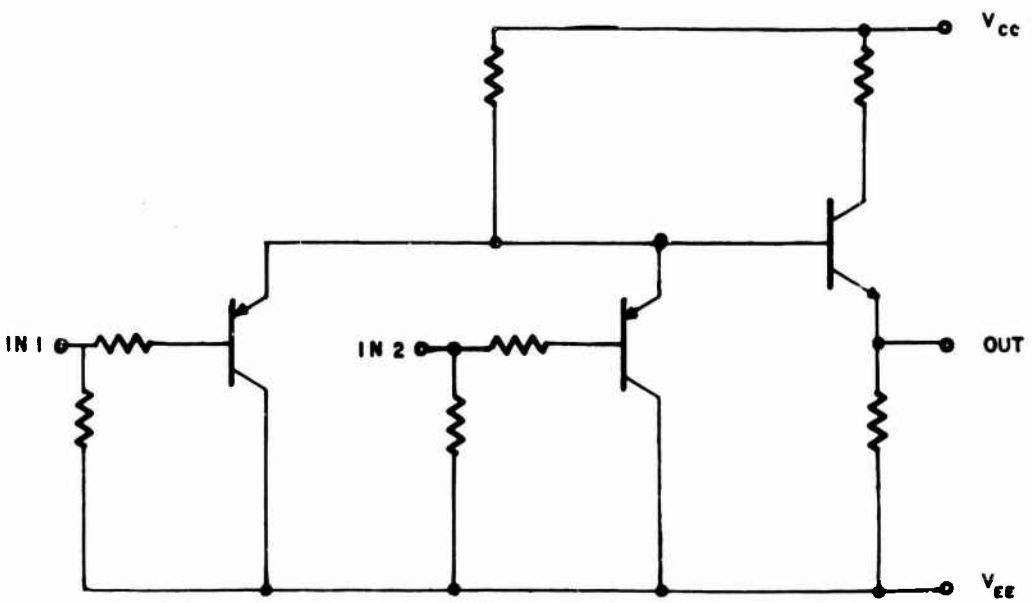


Figure 18. Fairchild CTL II Basic Gate Circuit

version of Fairchild's CTL family, which has been on the market for several years. The circuit differs from most other logic circuits in that it is not a threshold switching device, but an amplifier. ECL and TTL are examples of threshold circuits. When the inputs of a threshold gate begin to change state, the output remains static for a time. Then, when the inputs pass through the transition region of the gate, the output begins to move. Finally, after the input has passed through the transition region, the output is no longer affected by the input and continues to move until its transition is complete. Thus, the rise time of the input signal can be seen to affect the operating speed of a threshold gate, since the output does not move until the input signal has reached the threshold of the gate.

Inspection of Figure 18 shows that the CTL gate does not operate this way. The gate is basically a cascaded emitter follower circuit. When the inputs begin to move, a short propagation delay occurs and then the output begins to move. The significant difference between CTL and thresholded circuits is that the operating speed of CTL is not dependent on input rise time. CTL circuits are correspondingly designed to have short propagation delays and relatively long rise times. Figure 19 shows the relationship between input and output for a CTL II gate.

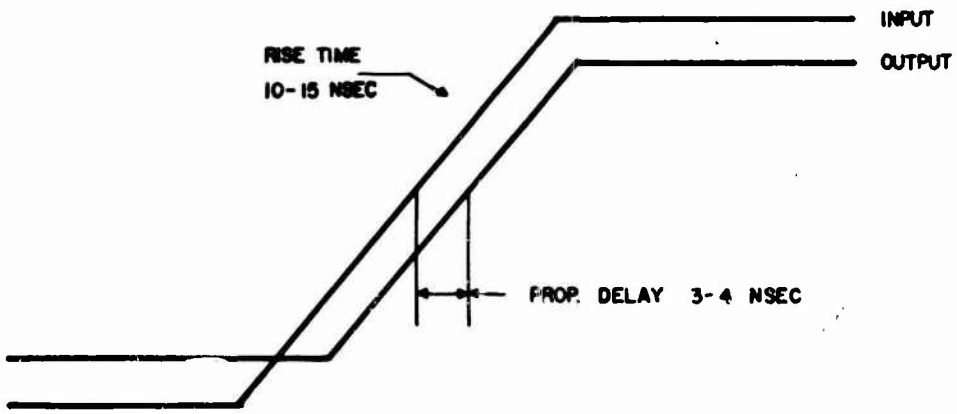


Figure 19. Relationship Between CTL II Gate Input and Output

CTL has several disadvantages. For one thing, the fact that the gain of a gate is less than unity means that the logic signal must get smaller and smaller as a signal is transmitted through several gates. Because of this, level restoring circuits must be used in the logic chain after a certain number of levels of logic have been used. Furthermore, the conventional definition of noise immunity does not carry much meaning when applied to CTL circuits. Ordinarily, one defines noise immunity for a gate as the number of volts an input can be moved (from either logic level) without producing a signal at the gate output. By this definition, the noise immunity of a CTL gate is zero. It is necessary to redefine noise immunity in terms of the actual application as the amount of noise voltage that can be introduced at a point in the system without producing an erroneous system output. Naturally, this definition is much harder to work with.

CTL does offer some significant advantages. Most of these relate directly to the designed-in slow rise time. Many of the difficulties encountered in signal transmission between parts of a system and in noise and cross-talk generated within parts arise from fast-transitioning signals. These problems contribute to both design complexity and to higher cost due to the necessity for using multilayer boards and coax for improved signal transmission capabilities. The non-thresholded nature of CTL allows the design of relatively fast logic without the necessity of having very short rise times. This advantage becomes more significant for very large systems.

3. Custom Bipolar MSI Programs

As was the case with MOS, there is a critical need for a means of obtaining specialized logic functions in integrated arrays, since manufacturers cannot afford to provide specialized functions as standard products. There are two methods by which such specialized functions can be obtained; by utilizing totally custom-designed circuits and by utilizing variable arrays. As with MOS, the variable array approach yields cost and time advantages for quantities less than 10,000 units.

Cost and time comparisons for custom and variable array approaches with bipolar circuits are roughly the same as those shown for MOS circuits in Table III.

Fairchild and Texas Instruments both have bipolar variable array programs in operation. Fairchild's program is called Micromatrix, while TI's is called Master Slice. There are significant differences between these two programs, but the similarities are numerous enough that only the Micro-matrix approach will be discussed here. This choice is not intended to imply that Fairchild's program is better than TI's; it is merely that the two programs are similar enough that only one need be described for illustrative purposes.

The 4500 Micromatrix is a cellular array consisting of eight identical cells arranged in a 4-by-2 pattern. Each cell contains four 4-input DTL NAND gates. The cells are interconnected by two layers of custom metallization to produce the customer's desired function. One characteristic of variable arrays is that only about half of the available silicon area is actually devoted to circuit elements; the rest is utilized for custom interconnections. In fact, the circuits themselves are much less efficiently laid out than would be the case with custom circuits, since considerable area must be used for the interconnection bonding pads. This less-than-optimum use of silicon area is responsible for the fact that variable arrays can never compete with custom circuits on actual production costs. The attractive feature of variable arrays is the time and fixed-cost savings they afford.

Fairchild also has available 4600 and 4700 TTL Micromatrix arrays, which contain six and twelve cells of TTL logic respectively. In both cases, each cell consists of four TTL AND-OR-INVERT elements. The layout is similar to that of the 4500 DTL array. Table VI compares these three different arrays. The AND-OR-INVERT elements are counted at two-gate-per-element complexity.

TABLE VI

COMPARISON OF FAIRCHILD 4500, 4600 AND 4700 MICROMATRIX ARRAYS

Array Type	Logic Type	Number of Gates/Array
4500	DTL	32
4600	TTL	48
4700	TTL	96

Propagation delays for the TTL elements are typically 8 nsec per gate for transmission on the chip and 12 nsec per gate for off-the-chip transmission. The corresponding on-chip and off-chip delays for the DTL gates of the 4500 array are 26 and 40 nsec for positive-going signals and 12 and 15 nsec for negative-going signals.

The most significant aspect of this variable array program is the degree of automation and computer-aided design assistance that has been applied to it. The entire process from the receipt of the customer's logic diagram through the delivery of master masks and test procedures to the semiconductor processing facility is handled by computer.

A portion of the computer aided design program simulates the customer's logic by means of a fairly extensive simulation program. Its purpose is to verify that the desired function is implemented and that the logic circuit will work when tolerances and delays are taken into account. Other functions of the program include the specification of the actual custom interconnection layers and the automatic generation of the required master mask set by means of a computer-driven plotting cutter. In addition, tests and test procedures peculiar to the function being implemented are generated for use after the arrays are fabricated. These tests are utilized with automatic or semi-automatic equipment to verify the proper functioning of the complete array. Manufacturers are careful to point out the importance of such test generation. Testing of complex arrays can be many orders of magnitude more difficult than testing individual circuits. The intelligent selection of tests and the incorporation of automatic techniques where appropriate is essential to reduction of the testing task.

Setup time required and costs vary considerably, depending on the type of function being implemented and on the manufacturer's work load. Initial cost of $10,000 and delivery time of six weeks would be a good estimate at the present time. Based on the utilization of silicon area, it would be expected that the production cost of a function implemented with a variable array should be about three times as high as the cost per circuit for a standard product of equivalent complexity.

4. LSI Programs

Texas Instruments discretionary wiring LSI program is the only program operational today that produces circuits qualifying, under practically anyone's definition, as large-scale integration. The basic steps in the

process are illustrated in Figure 20. Several different circuit types are fabricated on a $1\frac{1}{2}$ inch silicon wafer and the individual circuits are probed to determine which are good. The wafer probing is carried out at automatic probing stations, and the locations of good circuits on the wafer are recorded. The customer's logic diagram is punched into cards and is entered into the routing computer, along with the information on the location of good circuits. The routing program then determines a fully or partially optimized means of interconnection to provide the desired logical function with the known good circuits. The output of the routing program is a tape describing the metallization masks that are needed to implement the interconnections.

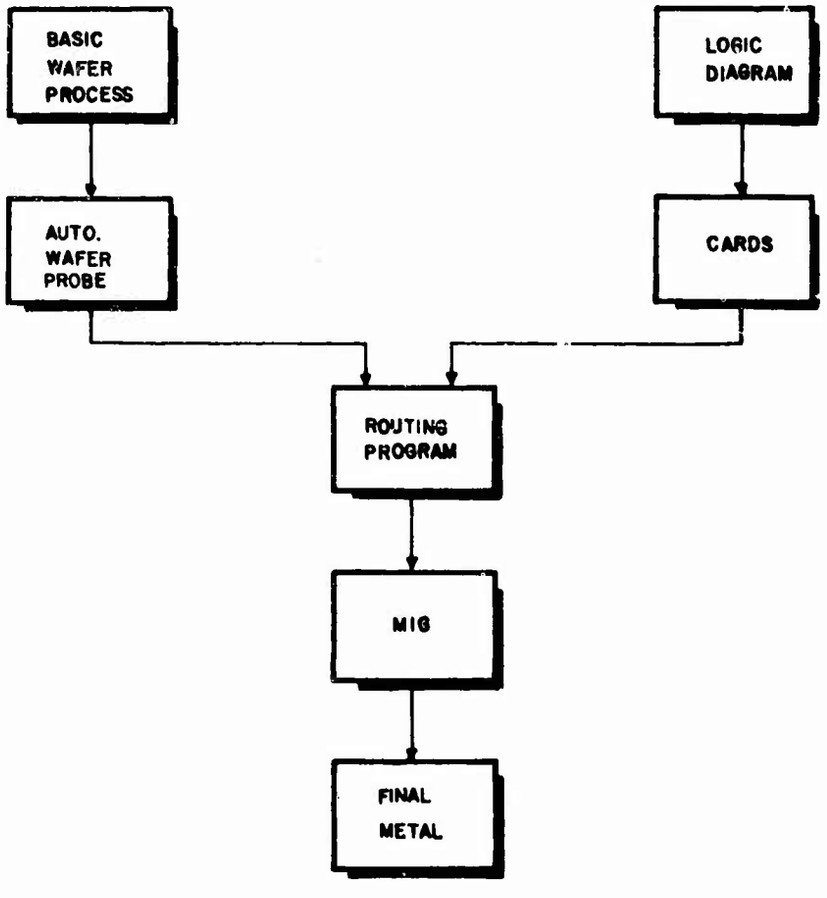


Figure 20. Steps in LSI Process

This tape is used to drive the Multilevel Interconnect Generator (MIG) System. The output of the MIG System is a high-resolution, high-accuracy, black and white CRT upon which the patterns of the required metallization masks are traced out. The patterns are recorded with a camera, the film from which is used for the actual metallization masks. Once the interconnection mask set has been made, the masks are used with the particular wafer to which they correspond to complete the circuit interconnections.

The significant way in which this Large Array System differs from any other and from MSI programs known to exist is that the desired functions are implemented from known good circuits. This allows using wafers that have less than 100 per cent yield to obtain working circuits.

The Texas Instruments LSI program allows a higher degree of complexity in arrays than any other program now available. This method has been used to implement a 16-bit general-purpose avionics computer for the MERA radar system. The computer is a redesign of the TI 2502 computer, which was originally built with 1735 integrated circuits. The LSI computer utilizes, instead, 34 discretionary wiring arrays of 14 different types.

Many objections have been voiced against the discretionary wiring approach to LSI, mostly by rival semiconductor manufacturers. Most of the objections boil down to "it won't work" or "it's economically unfeasible". Texas Instruments seems to have successfully met both of these objections, at least for the present, since hardware has been produced by the technique and Texas Instruments is offering the devices at competitive prices.

The circuits fabricated on the wafers are the same as those in Texas Instruments' Series 54 TTL logic family. Delay per gate is 13 nsec maximum, flip-flop shifting rate is 10 to 15 MHz, and power dissipation is 10 mw per gate. Two standard wafers are available. One of these, called the F slice, contains 1856 shift register stages, 96 clock drivers, and 48 buffers. The other standard is the K slice, which contains 128 JK flip flops and 642 gates. Not all of these circuits are good, of course. Yields at wafer probe appear to be about 70 per cent on flip flops and about 80 per cent on gates.

The process utilizes $1\frac{1}{2}$ inch diameter wafers. The finished array is packaged in a $2\frac{1}{2}$ inch square package having 39 leads on each of its four sides, for a total of 156 leads. Thirty of these leads are used for power and ground. The leads are on 0.050-inch centers. Power dissipation capability for the package is $2\frac{1}{2}$ watts.

The customer is not limited to F and K slices for the implementation of his function, although prices will probably be lower and delivery quicker with these than with slices of a non-standard composition. Theoretically, any mix of circuits from the TI Series 54 TTL line can be used to fill a wafer. This would include the possibility of utilizing the several Series 54 standard MSI circuits that are available. However, the farther the wafer design gets from standard, the longer the delivery and the higher the initial cost will be. Certain circuit configurations that have been used by TI in wafer composition are now called standard cells. A standard cell might be a full adder stage, for example, or a general-purpose counter stage. Wafers can be composed using standard cells without enormous increase in delivery time or fixed costs. If new standard cells have to be designed, however, the time and fixed cost required approach those of a custom integrated circuit - several months and several tens of thousands of dollars.

The primary limitation on the complexity that can now be achieved with the system appears to be the routing program. This program is constantly being improved. The difficulty of its task increases very rapidly with

the number of circuits or cells that are to be interconnected. If shift registers are to be constructed, the task of the routing program is relatively easy. Shift registers of 1000 bits per package (or several shorter registers per package with a total of 1000 bits) and with shifting rates of 10 to 15 MHz are fairly readily available. To implement random logic functions is much more difficult. The routing program now has a capability for interconnecting about 1000 nodes per circuit, where a node is an input or an output of a gate, flip flop, or cell. As can be seen, the complexity of the function that can be interconnected is greater if standard cells are used.

For shift register packages, the non-recurring cost is about $500 per package and production cost is about $1 per shifting stage. Random logic circuits involve a non-recurring cost of about $5000 and production cost of about $3 per gate. About 250 gates can be interconnected if the K slice is used, and about 500 gates can be accommodated through the use of standard cells. Delivery time would be about two to four months for a single piece and four to six months for fifty pieces.

A price of $3 per gate does not appear, at first, to compare favorably with the price of individual integrated circuits. However, such a comparison is misleading, since the gates in the LSI circuits are completely interconnected and packaged. Furthermore, the discretionary wiring LSI program is fairly new, and it is anticipated that production costs will decrease by a factor of three in the next year or two.

D. CRITERIA FOR VENDOR SELECTION

One of the most difficult choices to make when selecting vendors for semiconductor products is that between large and small manufacturers in the same product area. A small vendor will generally be more adaptable and more willing to make special-purpose circuits. He will usually be more profitable on low-volume items and therefore more willing to sell several different specialized circuits at reasonable prices. Any given number of dollars worth of circuits will represent a greater fraction of his total business than for a large manufacturer and will, therefore, be more attractive to him. A large manufacturer, on the other hand, will usually have more advanced technology in a given area. He will have more resources to back himself up in case of technical trouble and, since he cannot sacrifice dealing with any segment of the marketplace, he has a greater motive to preserve his reputation.

It appears that, since the dealings here are primarily with new technology in circuits and arrays, the advantages of greater technical backup would outweigh the advantages of greater flexibility. The decision would have to go to the large manufacturer, unless there were compelling technical reasons for doing otherwise.

VI. GENERAL PURPOSE COMPUTING SUBSYSTEM

A. GENERAL

The General Purpose Subsystem (GPS) consists of a configuration of standard computer units. It performs those functions which do not require the special purpose, high-speed computing capability included in the image generating subsystems. The GPS controls the image generating system; it contains the environment description, the view configuration and angles, and provides working data for the other subsystems.

The system computations are organized so that all of the data and software which must be changed from environment to environment is contained in the GPS. This simplifies the interface requirements with the remainder of the system and allows maximum use of off-the-shelf equipment for the computing tasks. The use of standard computing equipment also simplifies the software effort since programmers are more likely to be familiar with the assembly and higher order languages supplied.

B. COMPUTING TASKS

The computing tasks of the GPS fall into two categories, real-time and off-line. The real-time tasks occur during a simulation and include input and processing of flight data and computation of data for image generation. The real-time tasks are repeated each television frame time. The off-line tasks include maintenance, troubleshooting, environment construction and data formatting.

1. Real-Time Tasks

The following is a list of the main computations required during image generation:

a. Accept flight dynamics inputs from problem computer.

b. Calculate position and attitude of two moving vehicles.

c. Transform coordinates of L and P vectors to object and ground systems.

d. Perform "aspect" test to establish potential visibility of faces.

e. Compute the priority list.

f. Compute initial values for surface generating subsystem.

g. Compute point source image coordinates.

h. Compute air-to-ground weapons' trajectories and effects.

i. Transmit data to image generating subsystems:

1) Surface generating data
2) Point source data
3) Priority list to Object Calculating Section
4) I, and P vectors Object Calculating Section.
5) Color and control information to Object Processing Section and Video Processor.

2. Off-Line Tasks

The off-line tasks associated with constructing new environments include:

a. Accept and store vertex, face, and color data to describe the new environment.

b. Test environment data for proper format and consistency.

c. Assist in establishing priority relationships between objects and testing for proper environment.

d. Format and encode environment data.

e. Assemble operating programs.

The real-time tasks place constraints on functional characteristics of the computer such as instruction execution times and instruction repertoire. The off-line tasks are more demanding on storage because of the large amounts of data and the use of higher order assemblers and operating systems.

C. DESIRED CHARACTERISTICS

In view of the computing tasks during real-time operation, the following hardware characteristics are desired for the general purpose computer:

1. Binary word length of at least 30 bits.
2. Memory cycle time of one microsecond or less.
3. Two general purpose hardware registers and three hardware index registers.
4. Four quadrant hardware multiply and divide (fixed point); multiply time of 10 microseconds, divide time of 20 microseconds.
5. Direct addressing of the entire core memory.
6. Indirect addressing with post-indexing capability.
7. Immediate addressing of operands.
8. Logical and comparison instructions.
9. Shift operations (left and right) on single and double words.
10. Direct input/output channel.
11. Multiple port memory configuration.
12. List processing instructions.

The system should also have a proven software package which includes the following:

1. Fortran IV compatible with American Standards Association specifications.
2. Assembler for machine language mnemonics.
3. System executive for use with Assembler and Fortran compiler with full peripheral capability.
4. Utility routines for program modification and peripheral control.
5. Debug programs including conditional "snapshot" and memory dumps.

D. CONFIGURATION

A preliminary investigation of the computing requirements shows that the point source computation and the aspect and priority computation each require the major portion of a frame of central processor time. The configuration shown in Figure 21 shows how these requirements could be accommodated.

Inputs from the flight dynamics computer are brought in to CORE #1. The first central processor (CP# 1) processes these inputs to obtain dynamic vehicle positions and attitudes. The display parameters and location vectors are transformed to the appropriate coordinate systems and stored in CORE #2. The main task of the first central processor is connected with point source generation.

The second central processor is devoted to aspect testing and formation of the priority list for the Object Computing Section. The data for these computations are stored in CORE #2. The priority list which results from this computation is stored in CORE #3 which is shared with the Object Computing Section. This shared core memory also contains vertex data and L and P vectors related to object generation.

The main system peripherals are connected to CP #1 through its input/output processor. The necessary peripherals will depend to a large extent on the intended use of the image generation system. If it is to be used primarily for repetitious training over a standard library of environments, then a limited input/output system will suffice once the programs are established. On the other hand, if the environments are to be changed often or the system used for research purposes, then a more elaborate configuration is needed. A basic system would include a teletypewriter for each central processor and a paper tape input. If environments are to be modified fairly frequently a card reader and line printer should be added and the paper tape unit replaced with a magnetic tape unit.

The optional use of a disc is shown for rapid environment changes. This would interface with the second input/output processor for entering new object data directly into CORE #3. The disc would allow update of terrain generated by object techniques and be useful in extended missions where major portions of the object environment could be exchanged during a mission

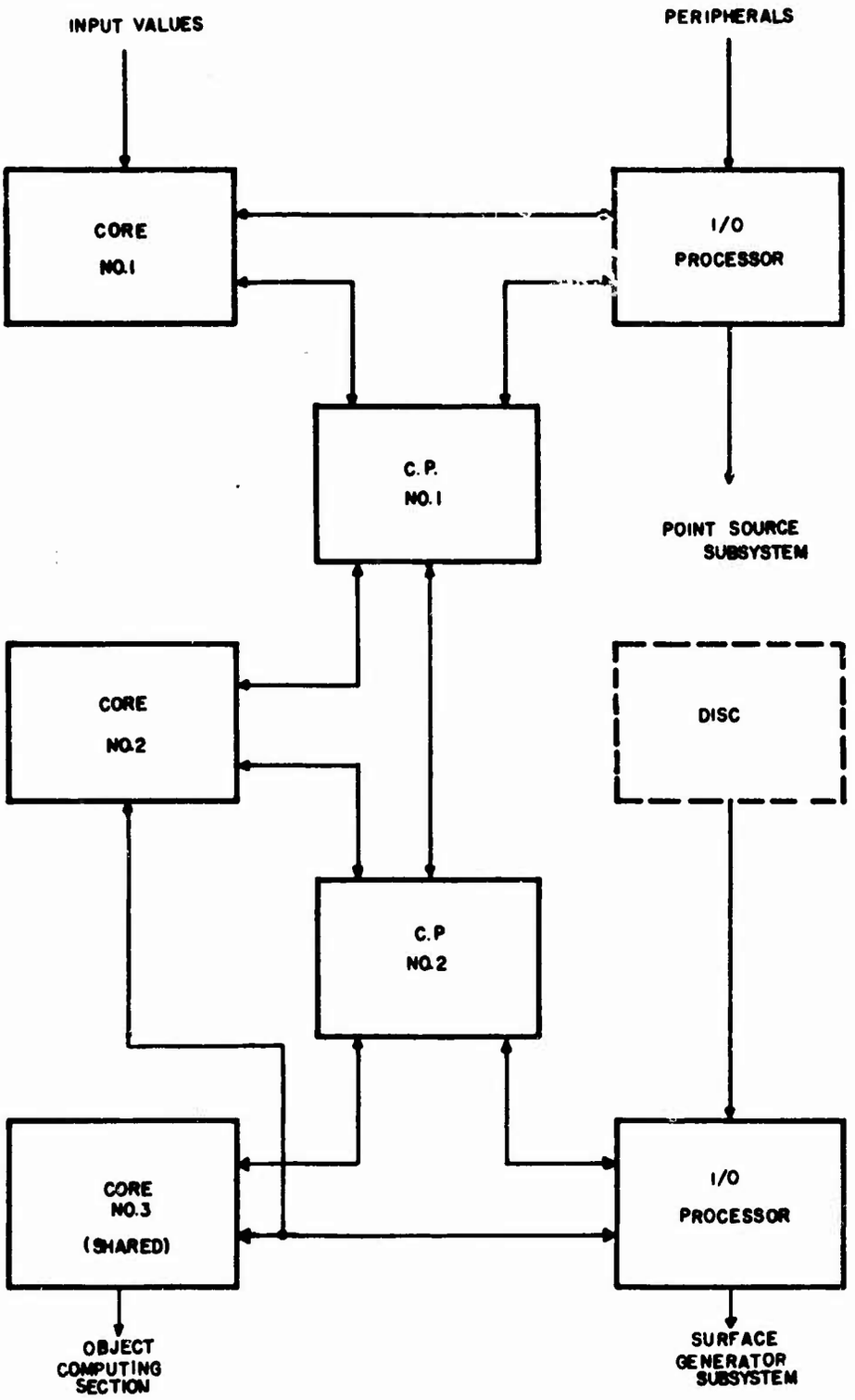


Figure 21. General Purpose Computing Subsystem

depending upon the geographic location of the aircraft. For example, a flight might start with takeoff from JFK (environment # 1), include an approach at SYR (environment # 2) and end with a landing at DAY (environment # 3).

The input/output processors contain control circuitry to permit data transfers from core memory without central processor intervention. The core memories are queried on a cycle-stealing basis and the program being executed by the central processor is not interrupted. The third core memory has a special interface described in Section VII. C. 1.

The size of the first two core memories are primarily determined by the off-line tasks of assembly and compilation. The storage required in the third core memory is determined by the object environment size. The following amounts of storage would be adequate:

CORE # 1 8000 words (point source data and off-line use)

CORE # 2 16000 words (face data and off-line use)

CORE # 3 8000 words (vertex and other object data)

E. COMPUTING TIME

The selection of the computing configuration and the division of computing tasks between the two central processors is based on preliminary estimates of computing time. Several of the more lengthy routines similar to those which would be used in this system have already been programmed on an SDS SIGMA 5 which has the required characteristics. This computer has a 32-bit word length and a memory cycle time of 850 nanoseconds. The multiply is executed in approximately 8 microseconds and a divide requires approximately 16 microseconds. Sixteen hardware registers are available and facilitate vector operations. Sine/cosine conversions, coordinate transformations and vector dot product execution times were measured and include time for address modification and memory cycle times.

On the basis of these measurements, the processing of input data and the computation of L and P vectors for two independent views would require:

12 sine/cosine conversions	1. 2 milliseconds
10 coordinate transformations	3. 0 milliseconds
overhead and other (est.)	2 milliseconds
Total	5. 2 milliseconds

The remainder of the frame time is available in CP # 1 for point source computations. The initial computations for a string of point sources requires approximately 0. 5 milliseconds; other points in the string require approximately 50 microseconds. Thus, the computation of 500 point sources is feasible only with a highly ordered set of lights such as on an airport.

When the point source generator is to be used for depicting tracer-bullet fire, the number of active (visible) rounds must be restricted. Approximately 50 active rounds may be computed with simple ballistics.

The second central processor performs the aspect test and forms the priority list. The arithmetic portion of these computations consists of repeated vector dot products which require approximately 50 microseconds each including related storage operations. One dot product must be executed for each environment face and for each separating plane. These tests must be repeated for the second view if it is independent. Assuming an average of four edges per face and two independent views, the aspect tests would require 12.5 milliseconds. The remainder of the frame time is available for priority calculations. A conservative estimate of the number of priority objects (and separating planes) is one quarter of the number of faces. Therefore, approximately 3 milliseconds are required for priority testing. Together, these two tests occupy one half of the frame time. The second half frame is devoted to list formation.

F. SOFTWARE

The real-time programs must be written so as to obtain maximum efficiency from the computing hardware. For this reason, all such programs are written for fixed point scaling and are originally coded in assembly language.

With the exception of certain portions of the aspect and priority programs, the real-time software is common to all environments. By the use of a few code words, control parameters may be easily changed to adapt the program to the different modes of operation required by the training tasks. The real-time programs would require on the order of 4000 machine language instructions.

The main off-line programs would consist of an object set-up routine and a data conversion and formatting routine. The set-up program would check new environment data for consistency, compute face normals, and examine priority relationships. It would require approximately 400 Fortran statements. The second routine would further process the output of the set-up program, convert data to fixed point scaling, and form the data blocks and lists for the environment description. This program is estimated to require 400 Fortran statements and several machine language programs. In addition to these two major programs, a number of service programs would be needed to assist in environment design. These would check adherence to priority rules, keep track of environment complexity, and provide other functions associated with environment design. The principal software development effort would be associated with priority processing.

VII. OBJECT COMPUTING SECTION

A. GENERAL

The Object Computing Section (OCS) performs the vector calculations associated with each object edge. The OCS contains the mathematical model of the environment which it processes in the priority sequence determined by the General Purpose Computing Subsystem. The OCS operates on the model with the display vectors for each view to produce the parameters that describe the edge images for the Object Processing Section.

The Object Generating Subsystem must generate 500 edges in two views at 30 frames per second. The frame time (33.3 milliseconds) and the total number of edges (1000) determine the required OCS computing speed. To a certain extent, the number of faces to be formed influences the design of the OCS (mostly the data and list storage requirements). In the worst case, all edges could be combined to form 332 triangular faces. This possibility requires that a three-edge face be computed in slightly less than 100 microseconds.

The display line standard to be used does not directly affect the complexity of the OCS. It simply implies that the computations be performed with enough resolution to limit the error in the edge images to less than the granularity of the display. Scaling for a nominal 1000 line display is assumed. The mathematical model must be stored in 30 bit words (minimum) to provide the required range and range resolution.

A portion of the OCS (the Normal Calculator) must process a number of edges equal to the number in the environment multiplied by the number of independent station points. If the two views were always juxtaposed (combined for one pilot) then a computing load of only 500 edges would be imposed. Independent views, such as for air-to-air combat, require a 1000 edge capacity. The complexity of the remainder of the OCS varies as the maximum number of edges per view multiplied by the number of views regardless of the view configuration.

The following are some of the factors considered in the conceptual design of the Object Processing Section:

1. Data must be processed for two display channels in the Object Processing Section.
2. The organization should not limit edge usage by assuming that environments possess certain simplifying properties.
3. Arithmetic operations should be controlled by fixed programs, not software.

4. Careful consideration must be given to the interfaces with surrounding sections to facilitate the numerous data transfers.
5. Where possible, a multiplicity of identical designs should be incorporated to reduce design cost.
6. Computation should be organized to avoid feedback paths and data-sensitive control functions.
7. Environment data must be easily accessible for rapid change.
8. High speed integrated circuits would be used because the logical complexity of the arithmetic units limit the application of presently available arrays.

B. ORGANIZATION OF THE CALCULATION

1. Normal Calculator

The viewing point is described to the environment by a vector from the viewing point to the origin of the environment coordinate system. This vector, L, is specified in environment coordinates as a triple quantity. Each vertex is described by a vector from the environment origin.

When a face has been selected for inclusion in the current display frame, the first vertex entry for that face is acquired. The position vector (L) is added to the vertex vector (V) to form a vector from the station point to the vertex (VL). Figure 22 illustrates these relationships for one face.

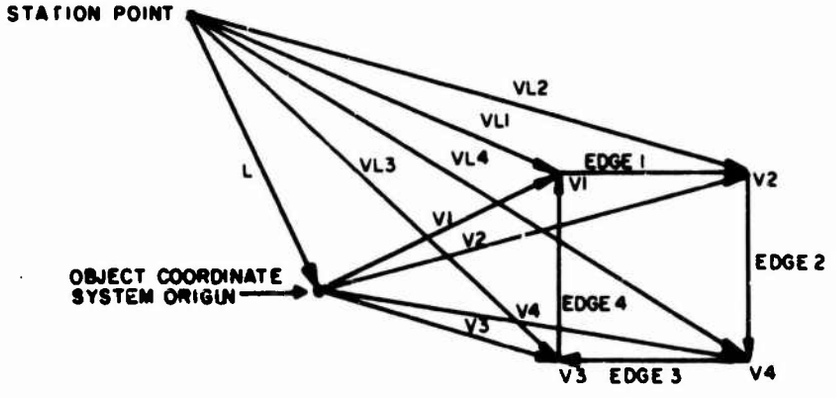


Figure 22. Vectors for Normal Computation

The vector cross product of VL1 and VL2 defines the plane containing VERTEX 1, VERTEX 2 and the viewing point. This plane is then defined by the vector that is normal to it:

$$\overrightarrow{VL1} \times \overrightarrow{VL2} = \overrightarrow{N1} \tag{19}$$

A subsequent calculation is employed for EDGE 2, EDGE 3, and EDGE 4.

Figure 22 also shows how each edge is related to pairs of vertices. For example, the values for EDGE 1 are derived from data for V_1 (vertex 1) and V_2 (vertex 2). The data for V_2 is used again, in defining EDGE 2, in combination with V_4. The data for this face are stored so as to facilitate sequential processing of the edges as follows:

FACE 1	VERTEX 1
	VERTEX 2
	VERTEX 4
	VERTEX 3
	VERTEX 1

The vector for the first vertex is repeated to close the loop for the last edge in the face.

The implementation of each vector cross product is then dependent on one "old" VL vector and one "new" VL vector. By buffering one triple quantity and calculating the next triple word value, each N edge face requires the computation of N+1 VL vectors. The cross product calculation requires a multiplication network plus a subtractor (or two's complement adder). The formation of VL would be accomplished by the same logic network.

For solid objects such as the hexahedron shown in Figure 23, the data

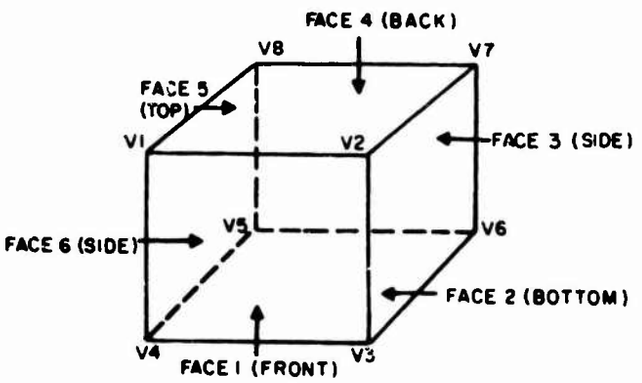


Figure 23. Hexahedron Designation

would be stored in the following tabular form:

LOCATION	VALUE
FACE 1 (front)	VERTEX 1 ↓ 2 ↓ 3 ↓ 4 VERTEX 1
FACE 2 (bottom)	VERTEX 6 ↓ 5 ↓ 4 ↓ 3 VERTEX 6
FACE 3 (side)	VERTEX 2 ↓ 7 ↓ 6 ↓ 3 VERTEX 2
FACE 4 (back)	VERTEX 5 ↓ 6 ↓ 7 ↓ 8 VERTEX 5
FACE 5 (top)	VERTEX 8 ↓ 7 ↓ 2 ↓ 1 VERTEX 8
FACE 6 (side)	VERTEX 1 ↓ 4 ↓ 5 ↓ 8 VERTEX 1

This table requires five vertex entries for each quadrilateral face and the eight vertices specifying the hexahedron require 30 vertex files.

For the attitude shown; FACE 1, FACE 3 and FACE 5 are oriented toward the viewer. Only these faces are represented in the data calculated by the OCS. An aspect computation performed by the GPS determines which of the faces are visible to each view. The GPS then furnishes a list which specifies the starting address for these faces.

A normal calculator samples vertex storage according to the list, forms VL vectors and normals. This section would process normals at regularly spaced intervals and supply them to the next section, the Edge Parameter Calculator. Control and color information is relayed to subsequent processing sections.

2. Edge Parameter Calculation

This section utilizes the edge plane normal (N) and the rotated display parameters (PC, PE, PL). The edge parameters (A, B) are the result of three vector dot products (QC, QE, QL) and two divisions. Each dot product uses the normal and one of the triple word display parameters. The display parameters are prescaled to allow fixed point computations. Each edge is matched to the appropriate set of display parameters that relate the viewer to the object environment. The divisions are independent of environment and viewer, but the resulting quotients must be transferred to the appropriate edge parameter buffer for each view. The edge parameters are transferred with control information describing color, the bounding nature of each edge (start or stop) and an indicator delineating the final edge for each face.

C. EQUIPMENT CONFIGURATION

The OCS is composed of three main portions; Vertex Storage, the Normal Calculator, and the Edge Parameter Calculator.

1. Vertex Storage and Interface

Vertex storage is implemented by a random access core memory which is shared with the General Purpose Computing Subsystem. This memory would have a cycle time of approximately one microsecond. An 8K (8192) memory would be required to accommodate an object environment for two 500 edge views. The central processor of the GPS can both read and write into this memory. This linkage facilitates loading of the environment data and the priority list.

Two interfaces are required for the vertex memory. They are shown in Figure 24. Both the GPS and the Normal Calculator must have access to the vertex memory. The interface with the GPS allows the priority list to be transferred during real-time operation. New environment data are also entered through this path when the environment is to be changed. Two priority lists must be held in the memory. One is a completed list for the current frame; the second list is formed as computed by the GPS and is to be used during the following frame.

A cycle-stealing technique would be used to permit insertion of the priority list entries for the following frame while the Normal Calculator is accessing vertex data for the current frame. A memory map separates data associated with the different views and absolute storage locations would be assigned to lists and display vectors.

When the system is operated in the off-line mode, the vertex core would be available for general purpose use. The interfaces with the GPS and Normal Calculator would require fully buffered address and data channels. The Normal Calculator controls memory accessing and addressing for vertex and priority list data.

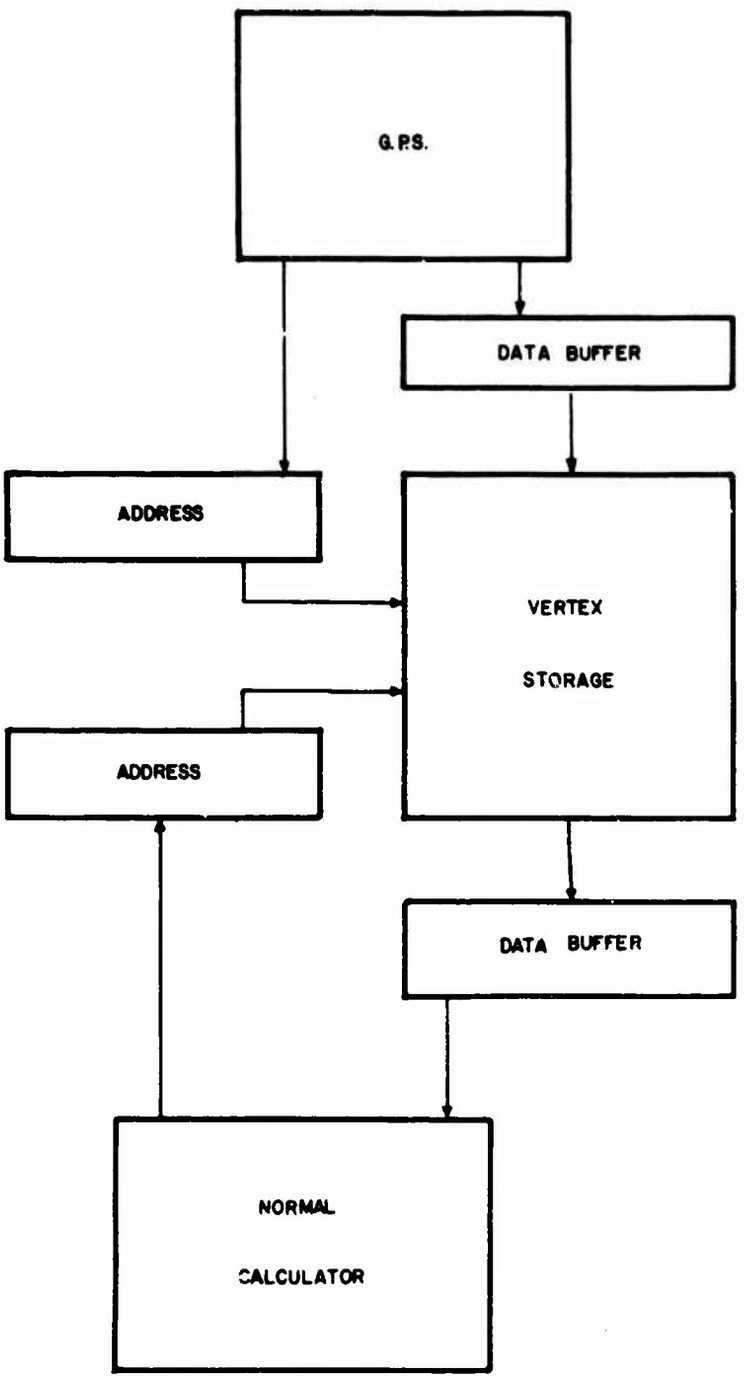


Figure 24. Environment Memory and Interface

2. Normal Calculator

The Normal Calculator is implemented as a triple parallel arithmetic function with memory addressing capabilities.

The edge calculation requires the following steps:

a. Obtain the x, y and z components from the Vertex storage by three consecutive fetches. These values relate to the "new" (or arrowhead) end of the edge.

b. Add the three components in triple parallel form to the appropriate "L" vector to calculate the VL vector for this vertex.

c. Normalize the vector.

d. Perform three of the multiplications associated with the cross product. The old (or arrow tail) VL value is the multiplier, the new VL value is the multiplicand. The multiplicand values are interconnected to provide linkage between the various components.

e. The three remaining multiplications are performed to calculate the other cross product components.

f. Terms are collected to form three vector components.

g. Resultant data is checked and scaled.

h. Triple word result is transferred to Edge Parameter Calculator.

i. The new VL value is re-located to represent the old VL value for the next calculation.

When a new face is begun, the process is altered to allow formation of an "old" VL value before any of the above steps are performed. The object is also matched to the appropriate L vector at this time.

As each edge is processed, it is checked for a control bit denoting that it is the last one for that face. When it is present, the priority list is queried to set the vertex address pointer to the location for the next face. As the edge plane normals are formed, this pointer is incremented with each data fetch.

The use of state-of-the-art circuitry allows the implementation of a carry-save multiply algorithm (for two's complement, 30 bit words) in 5 microseconds. If adds and subtracts are performed at sub-microsecond speeds, the process outlined for each edge could be accomplished within 15 microseconds.

A possible configuration for the Normal Calculator is shown in Figure 25. Only the X channel is shown; the other two would be similar. The central arithmetic unit and surrounding registers are shown.

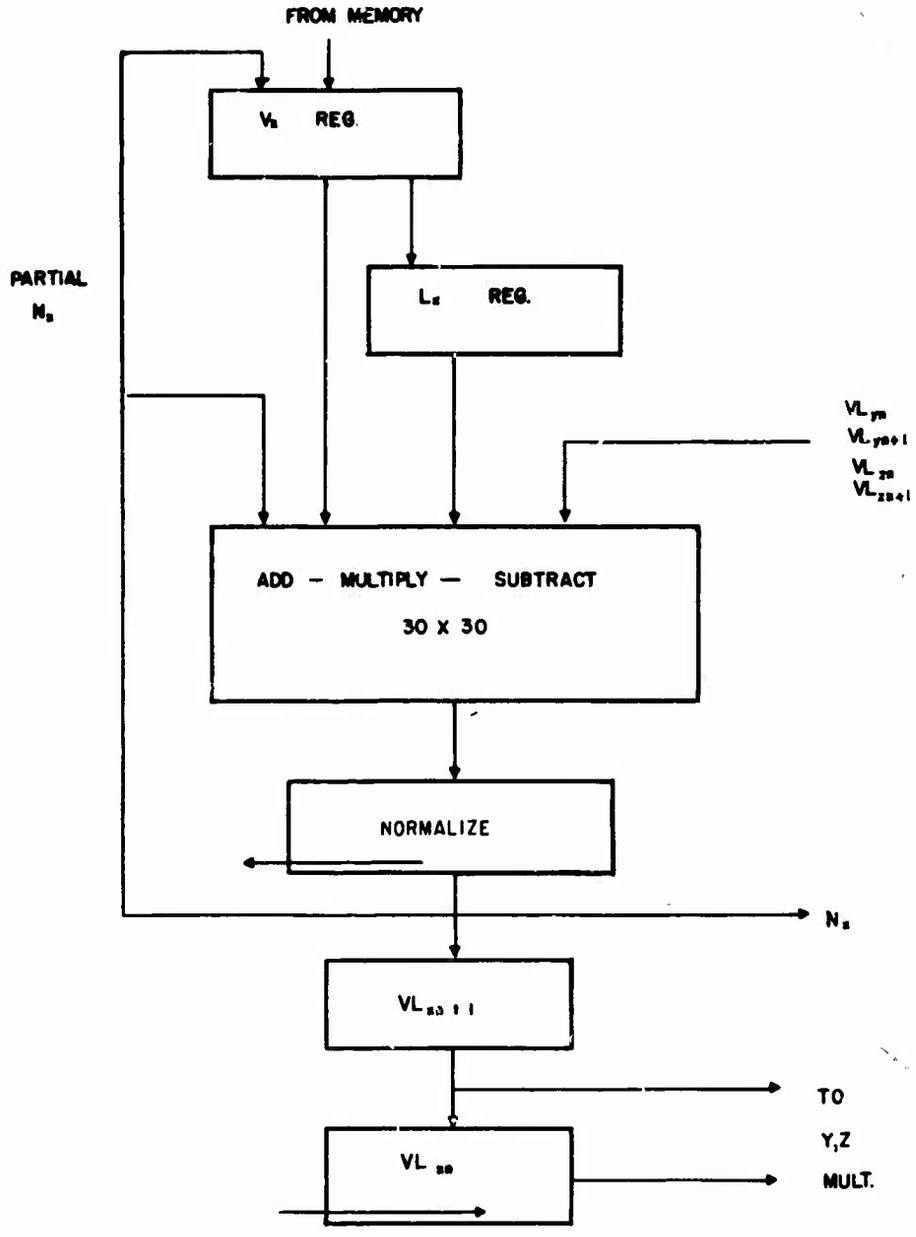


Figure 25. Normal Calculator (X Component)

3. Edge Parameter Calculator

The Edge Parameter Calculator (EPC) uses the edge plane normal (N) and the display parameters (PC, PE, PL) to calculate the slope and intercept values for each active edge.

The Edge Parameter Calculator is composed of two sections, a Dot Multiplier and a Divider. The EPC is organized to accept data as it is produced by the Normal Calculator. While the Normal Calculator is forming the values for the next edge, the Dot Multiplier is calculating the Q values for the current edge and the Divider is working on the data for the previous edge. As each new normal is received, the output of each section is presented to the next. The edge parameters (intercept (A) and slope (B)) are inserted into a buffer for the appropriate view.

a. Dot Multiplier

A triple parallel organization allows simultaneous computation of the three terms involved in the formation of a dot product. A serial adder structure combines these terms to form the resultant. A carry save algorithm identical to that employed in the Normal Calculator allows usage of identical designs. Each edge requires three dot products. The terms for one are combined, while the subsequent terms are being formed. The Normal (N) serves as the multiplier, while the appropriate triple value display vector is selected for the multiplicand.

Figure 26 shows the configuration for the Dot Multiplier. Display vectors appropriate to the view are obtained from the core memory. The normal vector components are held in buffer storage. The three multiplications are performed in parallel and the results are summed. The three Q quantities are formed sequentially and placed in buffer storage for use in the Divider.

b. Division

The second section of the Edge Parameter Calculator handles the division of the Q quantities to form B, A and A'. The block diagram of the Divider is shown in Figure 27.

The three Q values that were just calculated by the Dot Multiplier are selected as the divisors and dividends. For the usual slope value (B), Q_L is the dividend and Q_E is the divisor. This value is then tested to see if the edge is parallel or nearly parallel to the raster line. If it is, then a substitute value is presented for the B value.

If the B value denotes a slope with respect to the raster line, the usual A value is formed. This calculation employs Q_E as the divisor and Q_C as the dividend. The resultant quotient is again checked and, if necessary, a substitute value is presented. This is done so the values processed by the OPS can be contained in a minimum length data word.

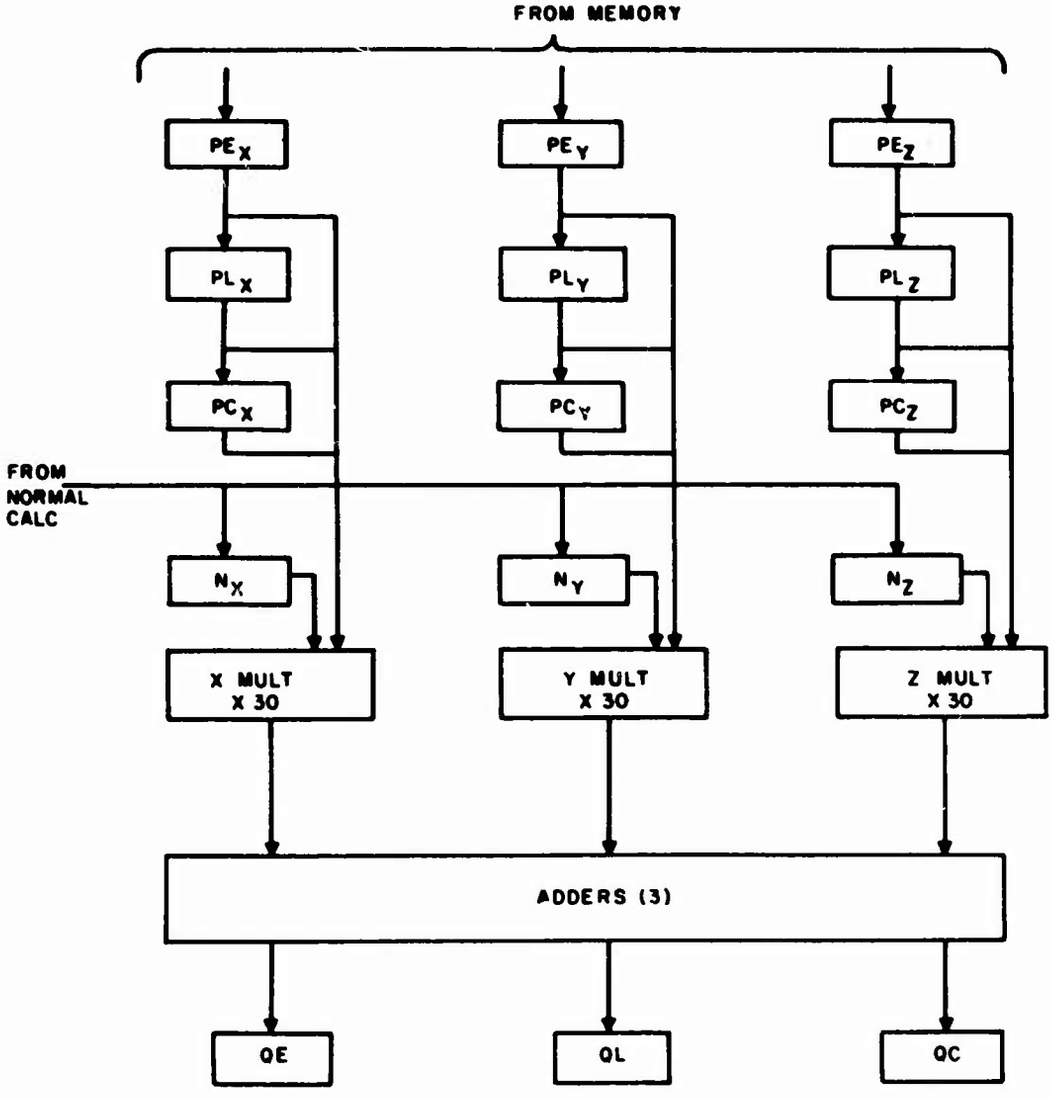


Figure 26. Dot Multiplier

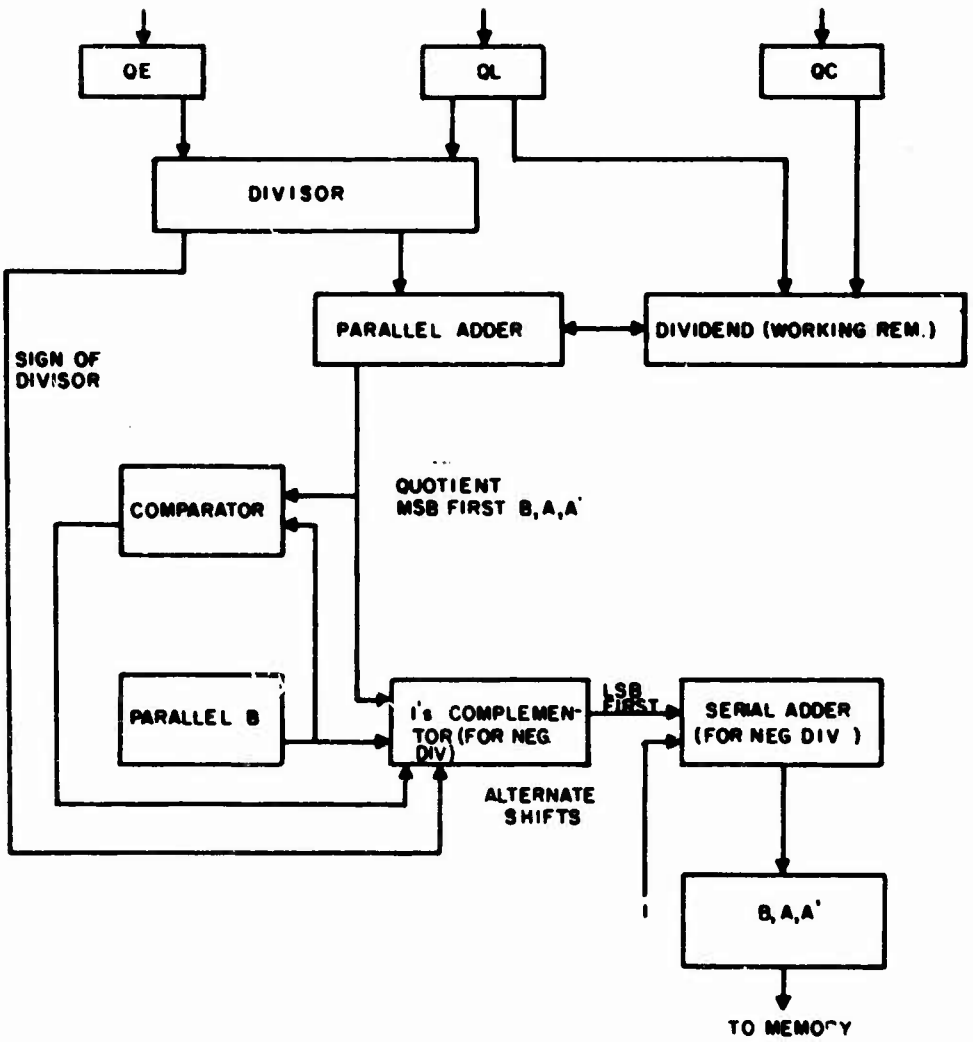


Figure 27. Divider

The implemented divide utilizes a parallel non-restoring algorithm. The first operation of this algorithm tests the relative magnitude of the divisor and the dividend. If the dividend is larger in magnitude than the divisor, an indicator is set to cause a maximum value to be used. This saturation value must have the same sign as the proper quotient would have had.

If the first division showed that the edge was parallel or nearly parallel to the raster line, a substitute A is formed. This quotient (A') is calculated by dividing Q_C by Q_L. Again the resultant value is checked and a substitute value is presented if necessary.

The edge parameters that are presented to the OPS also contain a control bit to denote the bounding nature (stop or start) of each edge. This bit is formed by the Divider Section and is included in the edge data transferred to the OPS buffers.

The color information originally extracted with the vertex data is restored to the appropriate edges at this time. The marker bit denoting the final edge in a face is also included.

VIII. OBJECT PROCESSING SECTION

A. SYSTEM FRAMEWORK SELECTION

The terms "system partitioning", "system definition", and "system architecture" all represent efforts to name a process that goes on early in a system design and which is usually vaguely defined at best. About the only general comment that can be made on all these terms is that, as systems have gotten more and more complex, the term applied has gotten more and more pompous. Here yet another term will be coined, in an effort to avoid preconceived notions attached to the old ones, and an attempt will be made to define it unambiguously.

The term that will be used is "system framework". System framework refers to a description of the computational tasks to be performed by a system, the relationship of these tasks to each other, and the means by which each task will be implemented. System framework selection begins with the specification of the inputs the system will receive and the outputs it will be required to produce. It proceeds with the design of the computation to be performed to produce the required outputs, with the constraint that the computation selected, from the many that would do the job, must be segmentable into subtasks that can be implemented with existing hardware. It ends with the definition of hardware blocks and their tasks and with the specification of the interaction between these blocks.

Before final system framework selection can be undertaken, an iterative process takes place in which the designer makes an approximation to the manner in which the computation will be carried out, then consults with hardware vendors to determine the feasibility of his first effort, and then goes back to modify his initial work in view of available hardware. This process may be repeated several times. A full knowledge of available hardware that is relevant to the computational tasks to be performed must exist before the designer can complete his task.

Selection of system framework does not constitute a full system design, because it does not include the determination of the feasibility - in terms of circuit delays, operating temperature, grounding problems, and so on - of every computational task within the system. What the completion of system framework selection does allow is the determination of the feasibility of the overall computation and the generation of a reasonable estimate of required hardware complexity and cost.

No process of system framework selection is ever fully complete. The optimal system framework depends on available hardware, and hardware characteristics are continuously changing - usually rapidly. What is attempted here is the selection of a system framework that demonstrates the feasibility

of the desired computations in the Object Processing Section (OPS), that allows a good estimate of hardware complexity, and that leaves room for adjustments or even considerable modifications to be made in the future, in view of changing hardware availability.

1. Ground Rules for System Framework Selection

The selection of system framework is governed by a number of constraints. The hardware available is one of these, as described above. The factors considered in selection of system framework are discussed below.

a. Hardware Availability

Many factors must be evaluated in determining the type of hardware - primarily circuits - that will be used. The type or types of technology to be used must be decided on; MOS, bipolar, or a combination of the two; individual integrated circuits, standard MSI arrays, custom arrays, available LSI arrays, or a combination. Technologies must be considered in terms of the complexity available in circuits of each type, the ease of application of each, and the amount of computing power obtainable per unit cost. In considering whether more than one technology is to be used, it is necessary to compare the resulting simplifications and cost advantages with the difficulties anticipated in interfacing and with noise and crosstalk problems due to different logic levels. Any cost savings must also be considered in the light of the cost of additional vendor interfaces. One estimate states that each major vendor interface on a large job costs $50,000. Even if the different technologies originate with the same vendor, it is likely that their engineering, production, and marketing functions are just about as separate as if they were in different companies. Finally, the cost of hardware familiarization for the designers must be considered for each previously unencountered technology.

Cost minimizations based on taking chances are a tenuous enterprise - they often turn out to be cost increases. The approach taken here has been to attempt to simplify the system where possible, rather than to try strictly to minimize costs. This does not mean that costs have been ignored; costs have been the overriding consideration. It does mean, however, that where a simple approach and a somewhat cheaper but more complicated approach competed for a task, the simpler approach was selected. This approach is appropriate at the system framework selection stage because it allows the possibility of cost reductions after more thorough investigation, while improving the likelihood that the selected approach will result in working hardware.

The same philosophy has led to the decision to base the system framework on available hardware rather than on anticipated advances in semiconductor technology. The example of the introduction of MECL III circuits described in Section V is partial justification for this approach. Expected advances have certainly not been ignored, however. A further criterion for a successful system framework has been that it must be adaptable to hardware advances when they occur.

b. System Parameters

The basic fixed quantities that the OPS must work with are the number of environment edges to be accommodated and the frame, line, and video rates desired in the display. The parameters that have been used for system framework selection are given in Table VII.

TABLE VII

SYSTEM PARAMETERS

Parameter	Value
Number of edges	500
Frame Rate	30 frames/sec (with two interlaced fields per frame)
Line Rate	1000 lines/frame
Video Rate	1000 active elements/line (⟶ 40 MHz video)

Although the above parameters have been used as the basis for the selection of framework, it has also been attempted to make the resulting system adaptable to more advanced system parameters in the future. For example, increasing the number of environment edges by a moderate amount can be accomplished by adding a block of hardware. It is not necessary to replace a large segment of the system.

2. Basic System Framework Approaches Considered

The following basic approaches have been considered:

1) Use of the fastest available circuits throughout.
2) Use of MOS circuits throughout.
3) Use of standard bipolar MSI products.
4) Use of custom MSI arrays.
5) Use of LSI arrays.
6) Use of combinations of the above approaches.

These approaches are discussed below.

a. High-speed Circuits

This approach actually implies the use of discrete integrated circuits, since complex functions and standard MSI arrays are generally available in medium speed logic lines but not in high speed. This is the case because the highest speed logic a manufacturer makes usually represents the

most complex processing he can do in terms of small geometries and close diffusion control. Therefore, he approaches the problems of increased circuit complexity in his medium speed logic line, which is much easier to make.

The high-speed circuit approach utilizes the smallest number of equivalent gates of any approach. However, this characteristic is not of much value since the cost per gate, packaged and wired into the system, is greater for this approach than for any other. The package count is maximized by this approach, which maximizes packaging cost.

In addition to the above-mentioned disadvantages, this approach stands a low chance of working at all, because of the great number of circuits, interconnections, and data paths involved.

b. MOS Circuits

The attraction of MOS circuits results from their low cost and from the large amount of circuitry that can be manufactured on a given silicon area. The first of these advantages makes it possible to utilize hardware freely to design around speed bottlenecks. The second helps to hold down system bulk, making intra-system communications less complicated.

The basic approach with MOS circuits is to use multiplexed operation and parallel computational channels wherever possible, rather than attempting to minimize package count, in order to achieve the required computing rate. In general, this is a viable approach. But it does produce considerable problems in two areas. The first area is in data handling problems that occur at several points where data format must be changed from serial, for multiplexed operation, to parallel, for parallel channel operation. The second is in control problems that result from the numerous modes of operation that exist and from the data-handling problems themselves. Both types of problems can be expressed by saying that a great deal of the computation can be handled by register-like structures, but only by changing from serial to parallel mode fairly often. The mode changing must be performed by and controlled by random logic. In addition, the serial and parallel operations themselves must be controlled by random logic. The net result is that the register-like structures do not compose the majority of the system.

The difficulties in using MOS circuits result from the logic and clock amplitudes, the limited driving ability of such circuits, and the unavailability of complex random logic functions in MOS. The logic swings of most MOS circuits are large - 10 to 15 volts peak to peak. For maximum operating speed, the clock swing must be greater than the logic swing - 15 to 25 volts. Both of these factors contribute to the generation of considerable noise, which could be a major problem with a very large system. Because of their high output impedance,MOS devices have limited driving capability. In fact, maximum shifting rates for registers of 20 MHz are possible, if no communication off the chip is required. But the chip output stage of such a register slows its operation to 2 MHz. This characteristic leads to the predominance of serial circuits - registers and a few counters, for example - in the MOS line, whereas serial-to-parallel converters with appreciable speed capability are practically non-existent.

The variable array MOS programs would go a long way toward providing the random logic functions that are needed. Unfortunately, these programs are in their infancy and are not sufficiently dependable to allow their use as a basis for system framework selection. Similarly, bipolar-compatible MOS circuits would simplify the noise and interfacing problems considerably, but they are not as yet generally available. It appears that the technological advances required to make an all-MOS system feasible should occur within the next year or two.

c. Standard Bipolar MSI Products

As mentioned in Section V, a considerable selection of medium-scale arrays is available from many manufacturers. A complete system could be constructed entirely with these elements. However, the interconnection problem, while not insurmountable as with the high-speed circuit approach, is a considerable obstacle.

d. Custom MSI Products

This category will include bipolar variable arrays and custom circuits made with both bipolar and with MOS devices. (MOS variable arrays were discussed above.) Bipolar variable array programs are operational. However, the complexity of circuits that can be implemented through these programs is still fairly low; they do not compete fully with standard MSI, in most cases. However, some specialized functions can very conveniently be implemented by this method. These arrays could serve as an adjunct to the use of other types of arrays.

Custom arrays, whether bipolar or MOS, involve an initial cost of some $50,000 and a delivery time of 9 to 12 months. Nearly three times the complexity of a variable array of the same size can be implemented. However, the cost per circuit with pro-rated fixed costs is greater than that for a variable array of equivalent complexity, unless the quantity required exceeds 10,000 for each type of circuit. On a cost basis, it appears very unlikely that the use of custom arrays would be feasible. If an attempt is made to construct a circuit that is very general and would fit in many applications, it is easy to negate the cost-per-circuit advantage by ending up with a would-be universal circuit that has much unused circuitry in it for most applications.

e. LSI Arrays

Texas Instruments discretionary wiring approach is a very powerful one. It offers both greater complexity and higher operating speed than any other type of array available, except custom bipolar arrays, which could be faster. An entire system could be constructed of these components, but such an approach would not be optimum in terms of cost.

f. Combinations

Several combinations of the above approaches have been studied, including combinations of standard, variable array, and custom MOS circuits, which are subject to the comments made in paragraph 2.b., MOS Circuits.

Combinations of bipolar and MOS circuits have been considered and have been rejected. It is felt that the cost advantages derived from the application of MOS circuits in the areas where they are readily used is outweighted by the problems associated with additional vendor interfaces and noise and interface problems.

3. Description of Selection Approach

a. OPS Tasks

The function of the OPS is to:

1) Accept data for each environment edge as projected into the display plane, in terms of slope and intercept quantities, at the beginning of each display frame,

2) Determine the intersection of each edge with each display raster line on a real-time basis,

3) Process the edges associated with each object face to determine which two edges bound each face on each raster line,

4) Implement the priority between faces so that each face properly obscures other faces,

5) Specify the gray shade (or color) of each raster element of each line on a real-time basis.

The content of these tasks is described below:

Figure 28 illustrates the function of slope (B) and intercept (A) quantities for a single edge as projected into the imaginary display plane. The edge itself is infinitely extending, but the only portion of interest is that segment which falls in the display plane itself. The edge is completely specified by its intersection A with the top of the display plane and by B, the amount by which its intersection with a raster line changes for each subsequent raster line. Note that both A and B can be negative.

The resolution along the raster line describes the fineness of the quantization with which a point along the line can be located - in other words, the number of pieces into which the line is divided for computational purposes. For this system, the line is divided into 1000 elements. So A represents the number of elements between the upper left hand corner of the display plane and the intersection of the edge with the first raster line. If B is added to A each time a raster line is scanned out, the cumulative sum will represent the distance in elements from the left side of the picture to the intersection for the line currently being drawn.

Every edge in the environment has associated with it an A and a B which represent it for the current frame. One of the tasks of the OPS is to add each B to its corresponding A during each raster line, so that the distances from the left side of the picture to all the intersections with that raster line will be available.

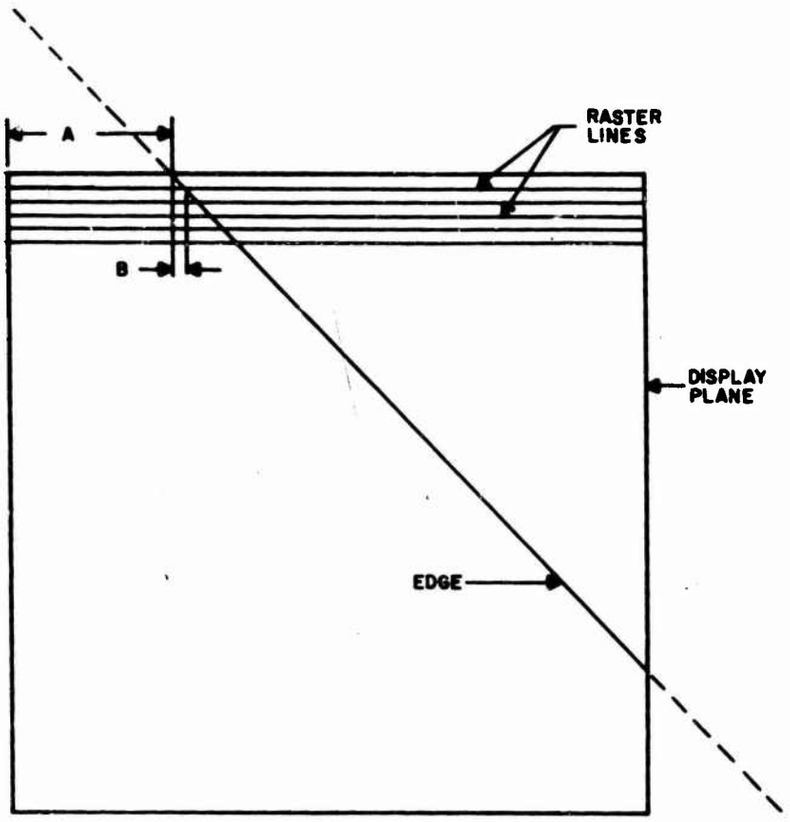


Figure 28. Slope and Intercept Quantities

When edges are processed by the Object Computing Section (OCS), all those edges bounding a single face are arranged in a contiguous group. A set of five bits accompanies the A and B quantities for each edge. One of these is an indicator bit that is set only in the last edge for each face. A second indicator bit shows whether the face occurs to the right or to the left of the edge as viewed on the display screen. A third is set for the special case where the edge is horizontal. The other two bits represent part of the gray shade information for the face in which the edge participates. The gray shade information for each face is described by a six-bit word. Since each face involves at least three edges, two of the six bits can be attached to each of the first three edges in a face group. These bits are accumulated later when they are needed.

The end-of-face (EOF) indicator bit allows the sequentially-received edge information to be broken up into groups that correspond to faces. Those edges having the face appearing to their right are called START edges, and those with the face to their left are called STOP edges. Several comparisons need to be made to determine what portion of each raster line, if any, intercepts a given face. Figure 29 illustrates this situation. The edges bounding the face A are numbered 1 through 4. Line k does not intersect the face at all. Line ℓ does intersect the face, and the bounding edges are edge 1 (a START edge) and edge 4 (a STOP edge). What characteristics of the edges can be used to determine which edges bound a face on a raster line?

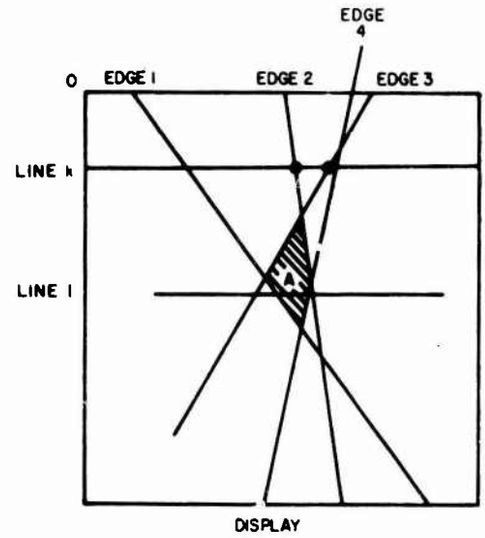


Figure 29. Face of Formation

For any face, one of the START edges intersects the raster line farther to the right than any of the others. This edge determines the left-hand boundary of the face on the raster line in question. (One can verify this by drawing a few faces and identifying the bounding START and STOP edges. Remember that faces must be convex polygons.) Similarly, the STOP edge that has its intersection with the raster line farthest to the left is the right-hand boundary of the face on the raster line. Of course, if the left boundary is to the right of the right boundary, the face STOPS before it STARTS and does not appear on the line at all.

Recall that the cumulative sum of A plus some number of B's is updated for every raster line. Each of these sums represents the intersection with the raster line of the edge with which the sum is associated. To determine which START edge bounds the face, then, the intersections (cumulative sums) for all the START edges for a face are compared and the largest one is selected. Finally, the selected START is compared with the selected STOP to see whether the face appears on the raster line or not. The intersections of START edges with a raster line are called START numbers. STOP numbers are similarly defined. The comparison of START and STOP numbers to determine the extent of the intersection of a face with the raster line is another of the tasks of the OPS.

The groups of edges representing faces are received from the OCS in priority order. The face or faces that are not wholly or partly obscured by any other faces are received first; and then the faces that these obscure wholly or partly are received, and so on until the last face that everything can obscure is received. The receipt of these faces in priority order allows the implementation of priority between faces when the lines of video are prepared for display. This implementation is carried out by determining the gray shade of each element of a raster line and by storing the six-bit binary code for that element's gray shade in an array that has a storage location for each element in a raster line.

Figures 30 and 31 describe this implementation. A hypothetical display plane is shown which has 40 elements and 40 lines. The image of a tetrahedron is to be drawn, and this image is composed of two visible faces,

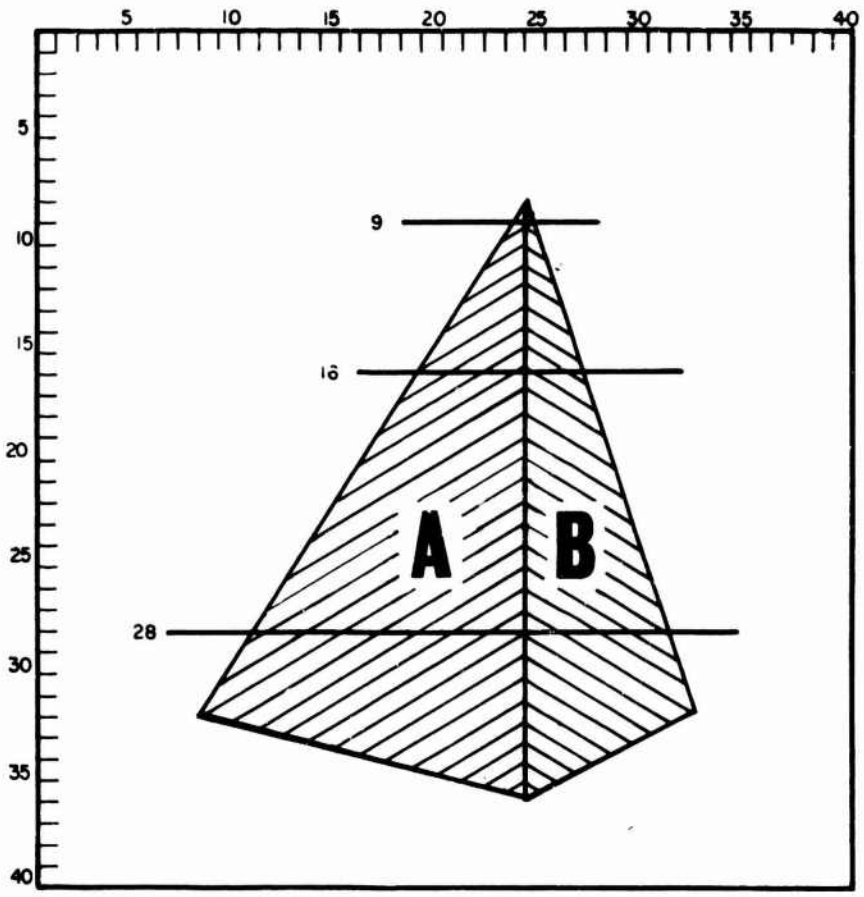


Figure 30. Image of Tetrahedron

face A and face B. Figure 31 shows what gray shades would be stored in what element numbers for three sample raster lines: 9, 16, and 28. For instance, when line 16 is being composed, the contents of all element storage positions would initially be set to some background shade. Then the shade of face A would be loaded into locations 20 to 24 and the shade of B would be loaded into locations 25 through 28.

Suppose now that some other face C is to be loaded and that it has lower priority than faces A and B. Figure 32 illustrates such a situation. Since the faces are received and processed in priority order, faces A and B will be loaded into the array before C on every line where all three occur. Whenever a location in the array is loaded, the access to it is inhibited until after the raster line being composed has been displayed. Thus nothing can be loaded in over what is already there, unless what is already there is the initial background color. On line 16, for example, A would be loaded from 20

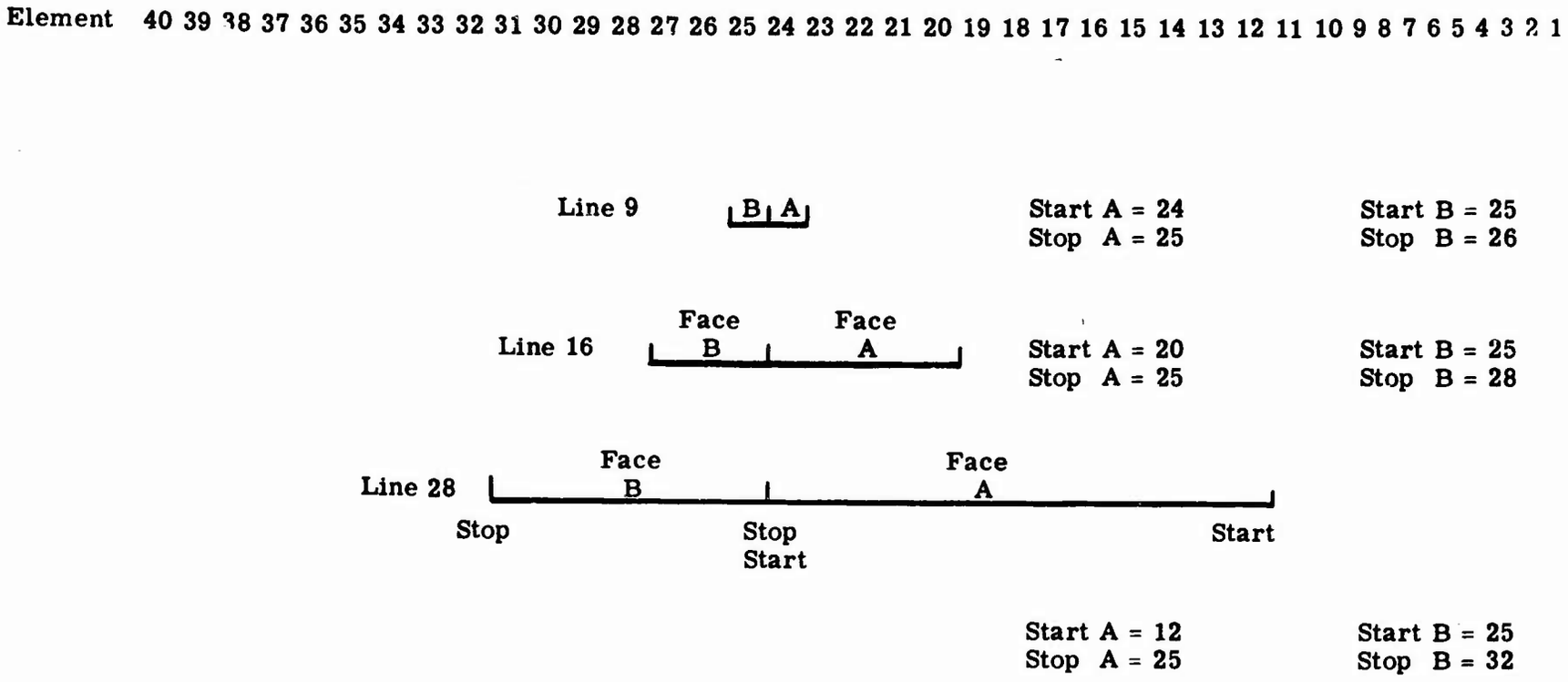


Figure 31. Video Assembly Register Operation

through 24 and B would be loaded from 25 through 28. Then C would be loaded, but only from 29 through 36. Elements 26 through 28 have already been loaded with B and therefore are not accessible.

Once all the faces appearing on a raster line have been loaded into the array, the contents of the array are read out sequentially, in synchronism with the scanning beam on the CRT face, and the 6-bit gray shade codes are converted to analog numbers to specify the shade of each element on the line. This process is repeated for each line to provide the entire picture. This line composition function is the final task of the OPS.

b. Block Diagram

The flow of information through the OPS is shown by the block diagram of Figure 33. The units of this diagram correspond to the functions described above.

The Edge-Parameter Buffer accumulates the edge slope and intercept information, along with the appended control and gray shade bits for each edge, as the information is generated by the OCS. At the end of a frame, a complete set of new edge information has been accumulated. This information is transferred to the Edge Storage and Update Unit during the vertical retrace interval. The Edge Storage and Update Unit performs the A-to-B additions during each raster line and presents the cumulative sum for use in determining face intercepts.

The Face Determination Unit performs the comparison of START and STOP edges for each face to determine the extent of each face on each raster line. The START and STOP numbers bounding each face are determined during each raster line.

The gray shade code for each face must be loaded into the Raster Line Composition Unit for all element positions between the pertinent START and STOP numbers, except where inhibited by higher priority faces already loaded. This loading is mechanized by the Face Loading Unit. The Face Loading Queue is utilized to average the rate at which faces are to be loaded over the period of the raster line.

After an entire line has been composed, the stored results are read out of the Raster Line Composition Unit in synchronism with the scanning beam, and the gray shade codes are converted by the D/A Converter Unit to analog voltages for application to the Display.

c. Characteristics of the Selected Approach

Several criteria for a successful system framework were set down earlier in this section. A discussion is pertinent, at this point, as to how well the selected approach meets these criteria.

One main consideration has been to make the system reasonably simple in terms of operating speed and . These two ends work against each other in a way, since operating speed can generally be increased by using more equipment. The compromise approach taken has been to utilize

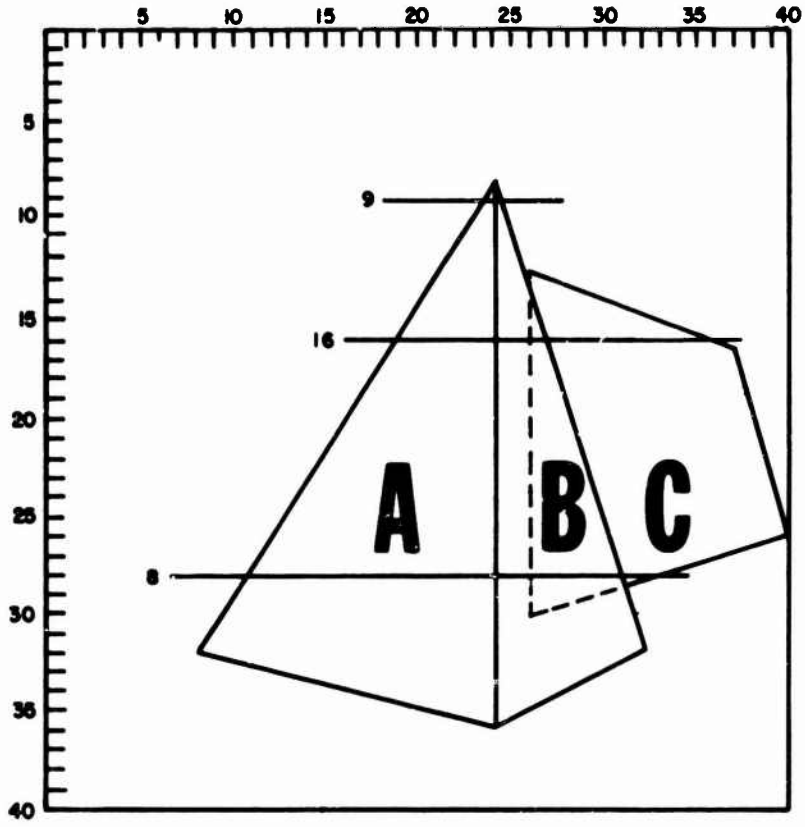


Figure 32. Partially Obscured Face

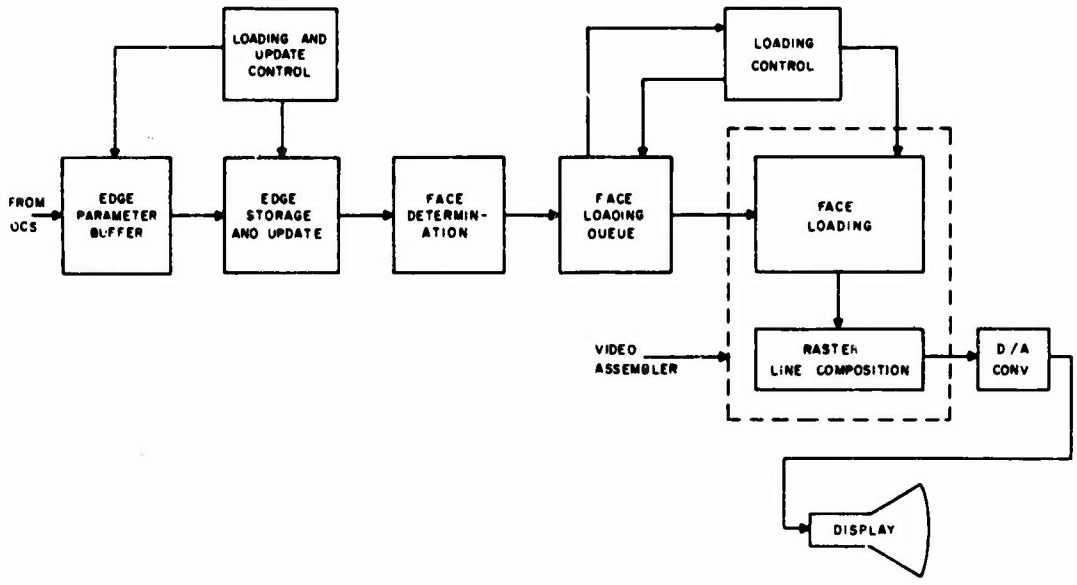


Figure 33. System Framework Block Diagram

sufficient parallel and multiplexed operation to keep clock rates significantly below the maximum permissible for the logic type used. To achieve this, the only part of the system that does not employ multiplexing or parallel operation is the Face Determination Unit. This Unit is a relatively small part of the system and, although the data rates required within it are high, preliminary work indicates that the design will not be difficult.

The estimates of hardware complexity contained in Section XI show that the overall bulk of the OPS as proposed permits its containment in a single large cabinet. This limitation of size is certainly more than sufficient to simplify the intra-system communications of the OPS. Possible problems with data communications and system noise are further simplified by the use of a single logic family - TTL - throughout all the OPS except the D/A converter.

The system framework for the OPS is based entirely on hardware that is available now. However, in line with another criterion for the framework, the system has been made adaptable to hardware advances when such become available. Some bipolar MSI circuits would be used in the system as the framework now stands. Other parts of the system are implemented with individual integrated circuits; these parts could easily be adapted to bipolar variable array implementation when this technology reaches a sufficient level of complexity. In addition, the selection of TTL logic allows changing the implementation of much of the system to MOS when bipolar- compatible MOS circuits become available. So, depending on the time scale for the completion of the system design and the beginning of the construction phase on a large system, the impending advances in MOS circuits could allow the implementation of significant parts of the system with MOS where cost savings would result.

Adaptability to advanced system parameters was another criterion for a successful framework. The following description of the proposed implementation will show that the desired modularity of the OPS exists in terms of environment edges and number of elements per raster line. Furthermore, the system could be converted to full color operation by the incorporation of a color memory unit, which will also be described.

B. DESCRIPTION OF SYSTEM FRAMEWORK

The individual units of the OPS are discussed below in terms of their implementation. For each segment of the system, the hardware technologies to be employed are specified, and the internal computational rates and data rates at interfaces are discussed. The manner in which the data and computational rates depend on system parameters is stated. Those parts of the system are pointed out in which bipolar MSI and MOS circuits could replace existing technology.

1. Edge Parameter Buffer and Edge Storage and Update Unit

Figure 34 shows the Edge Parameter buffer and Edge Storage and Update Unit. The Edge Parameter Buffer receives an edge work (A, B, and gray shade/control bits) for each environment edge during each frame. These edge words are accumulated during each frame until a complete set has been

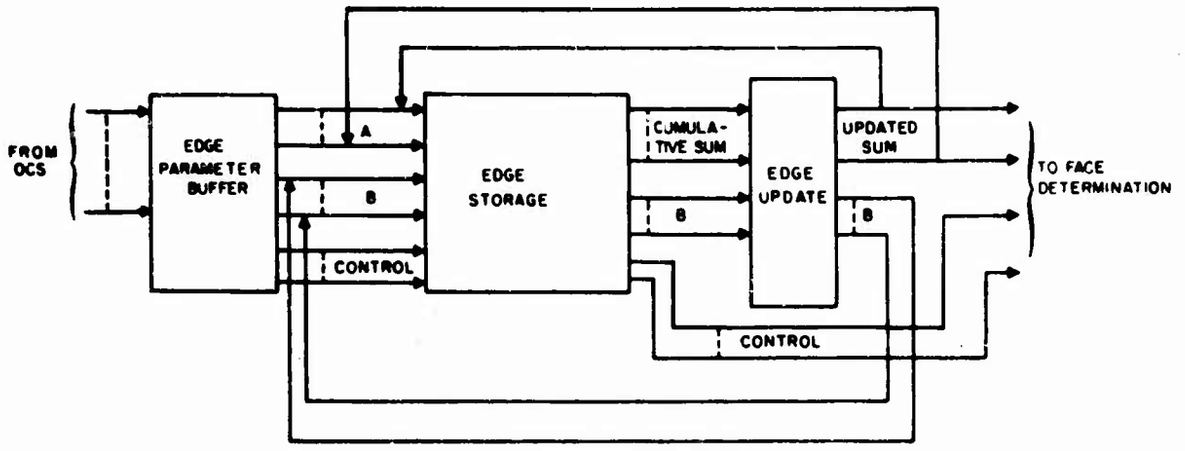


Figure 34. Edge Parameter Buffer and Edge Storage and Update Unit

received, at which time the Edge Parameter Buffer contents are transferred to the Edge Storage and Update Unit for the generation of the next frame's picture. The edge words will be received in parallel. The rate of receipt of words is seen to be fairly low - 33 msec/500 edges = 66 μsec/edge for the parameters assumed. The rate at which the information must be read out to the edge storage equipment during vertical retrace is 20 to 30 times higher, but still allows at least 2 μsec per edge. A commercial core memory will be used for the Edge Parameter Buffer.

The most difficult requirement for the Edge Parameter Buffer to meet is the data rate that occurs when the accumulated edge words are transferred to the Edge Storage and Update Unit during vertical retrace. This data rate is proportional to the number of environment edges employed. Core memories having cycle times of 1 μsec or less are fairly readily available, so a considerable increase in the number of environment edges could be accommodated by using a fast memory. When this approach has been extended as far as possible, core memories can be interleaved to achieve higher data rates. With a 1-msec vertical retrace, about 3000 edges could be accommodated, by using interleaved commercial core memories, without going to highly complex multiplexing schemes.

The Edge Storage function will be implemented with shift registers and applies the cumulative sum and the B quantity for each edge in parallel to the Edge Update adder. The shifting rate of the registers is limited to about 2 MHz, in order to allow a 500-nsec period for the sum to be formed. Both the Edge Storage and Edge Update functions will be implemented with Texas Instruments LSI arrays. The 2-MHz shifting rate is very conservative, compared to the 10 to 15 MHz capability of the array circuits. Preliminary estimates indicate that it should be possible for a sum to be formed in about 300 nsec, so considerable margin exists in the adder too.

The edge cumulative sum must be updated each line time. Since 500 nsec is required for each update and the line time is about 32 μsec, the module described would be able to accommodate 64 edges. ; Thus, eight levels of multiplexing will be required to allow processing of $8 \times 64 \doteq 500$ edges during a raster line period.

The multiplexing functions in the Edge Storage and Update Unit are performed by standard TTL MSI circuits. The control and miscellaneous functions will be performed by a combination of standard MSI and individual integrated circuits.

The Edge Storage function and its associated control and multiplexing circuits form a likely candidate for MOS implementation, when faster, bipolar-compatible circuits become available. The registers within the Edge Storage function could probably be implemented with standard products (a maximum shifting rate of 3 to 4 MHz would be required to provide some margin), and the remaining parts could be fabricated by means of a variable array program. The adder itself is unlikely to be feasible in MOS, since the speed requirements on it are quite stringent and the complexity and degree of specialization are high.

If more edges are to be accommodated within a line time, additional levels of multiplexing can be added. The amount of hardware required is more or less proportional to the number of edges.

2. Face Determination Unit

The Face Determination Unit performs the comparisons between START numbers and STOP numbers for each face, to determine which pair bounds the face on the raster line being processed. There are three comparators involved: one to find the greatest START number, one to find the smallest STOP number, and one to determine, after the appropriate START and STOP have been found, whether the START is less than the STOP. In addition, an accumulator is needed to assemble the three two-bit segments of the gray shade code, which are appended to the first three edges associated with a face, into a single 6-bit word.

The Face Determination Unit receives a new input every 64 nsec during a raster line. Since at least three edges participate in each face, 190 nsec is available for a single face decision. High-speed TTL circuits will be used to implement this unit. Preliminary investigations indicate that the comparators will require three or four levels of logic. With gate delays of 6 to 7 nsec, this implies that a comparison can be completed in no more than 28 nsec, which is well within the 64 nsec available. The Face Determination Unit is a relatively small part of the entire OCS. It will probably require about four cards of logic.

The Face Determination Unit is capable of handling about 1000 environment edges in its present form. Higher numbers of environment edges can be accommodated by utilizing parallel comparators. This technique is illustrated by Figure 35. Four serially-received quantities are compared in groups of two and the results of these comparisons are compared to determine the final result (largest or smallest of the group). As soon as the

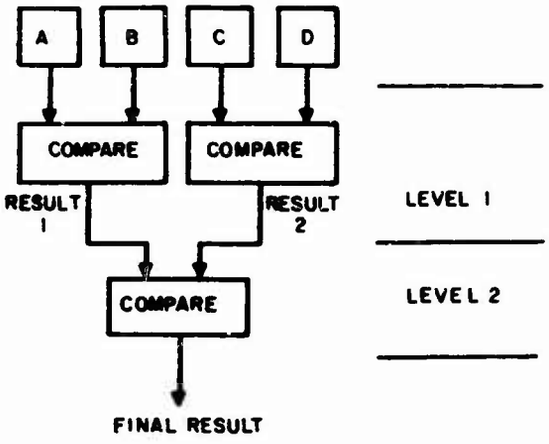


Figure 35. Parallel Comparison

first level of comparisons - that between the groups of two - is complete, another set of four quantities can be entered. Thus, one set of quantities can be compared in level 1 while the results from another set are being compared in level 2. Two levels of comparison must occur during the time required for four input quantities to be received. This approach can be used to achieve greater speedup by using more levels of comparison. Three levels of comparison will find the largest or smallest of a group of eight, for example.

It is unlikely that MOS could be applied in the Face Determination Unit because of the high operating speeds required. However, standard MSI products that could be used to implement the comparisons might very well become available in the near future.

3. Face Loading Queue and Video Assembler

Figure 36 shows the units involved in the line-by-line composition of video for the display. The inputs to these units are the bounding START and STOP numbers for each face on a raster line and the 6-bit gray shade code corresponding to the face. Recall that the Raster Line Composition Unit has a storage location for each element in a raster line. The desired end result of the loading of START, STOP, and gray shade information associated with the faces is to have each element location loaded with the gray shade of the face, or part of a face, that is to be visible on that element in the display. The Face Loading Unit accomplishes this by assembling the face shades as they are received in priority order, as described in Section VIII.A.3, until an entire raster line has been assembled. Then the stored description is transferred directly into the Raster Line Composition Unit, from where it is shifted out serially to the D/A Converter Unit. Thus, the Video Assembler contains two sets of storage elements: one for the raster line being assembled, and one for the line being displayed. Each of these stores has a six-bit location for each element in a raster line.

With 500 environment edges, the maximum number of faces that can occur is 166 (achieved by making the maximum possible number of triangles).

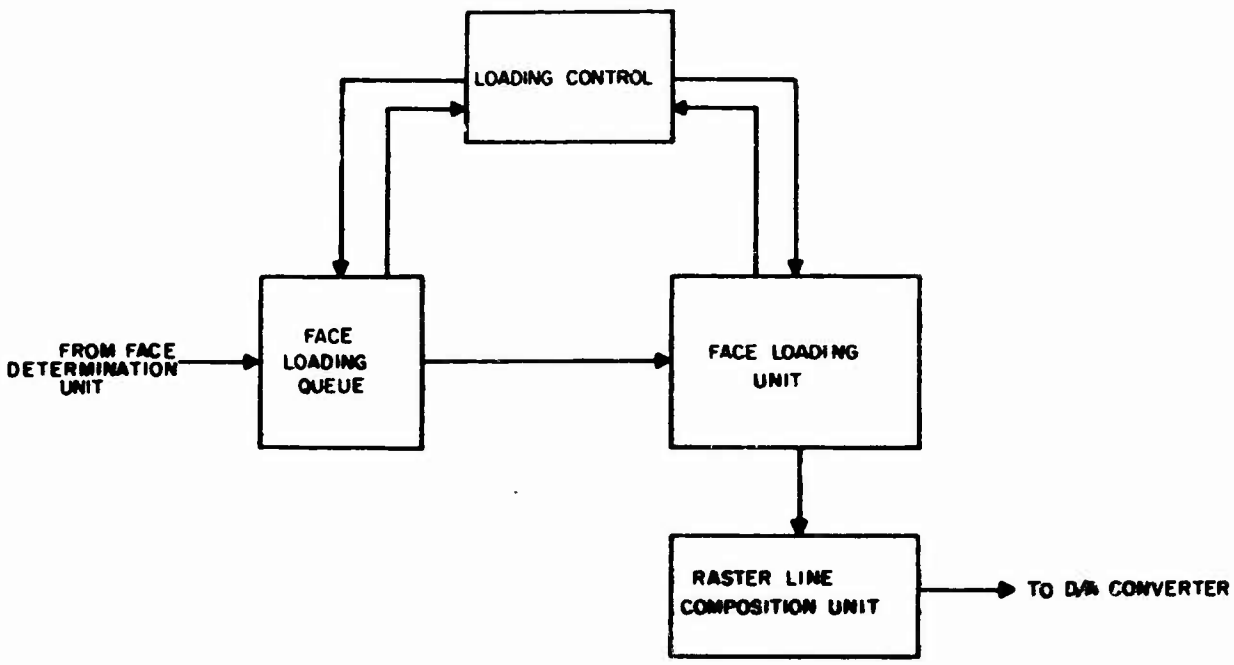


Figure 36. Video Assembler and Queue

With a 32 μsec raster line period, the average amount of time available for loading a face is 200 nsec. The inputs from the Face Determination unit will arrive at odd intervals, with the maximum average rate being one face every 200 nsec. The Face Loading Queue smooths the data rate to the average, so that the Face Loading Unit will not have to accommodate the peak face rate. If the Face Loading Unit can be designed to complete the loading of a face in the minimum time that could ever occur between receipts of sets of face information, the Queue would not be necessary. The Queue is a relatively small piece of hardware, and its presence or absence will not have a great effect on system complexity.

The Face Loading Queue, if necessary, will be implemented with standard bipolar MSI products - primarily 64-bit random access memories. Texas Instruments LSI will be used for the Face Loading Unit, and the Raster Line Composition Unit will be constructed of four-bit universal registers, an MSI product. The Loading Control equipment will be composed primarily of high-speed TTL integrated circuits.

The information is transferred from the Face Loading Unit to the Raster Line Composition Unit at the beginning of each line. This transfer is accomplished by shifting the six-bit numbers between the two units. All the six-bit numbers are available simultaneously, and the bits of each number appear sequentially. Thus six shifting periods are required to accomplish the transfer.

After the information has been entered into the Raster Line Composition Unit, the gray shade codes must be shifted out sequentially for display. This time, all six bits of each gray shade number must be available in parallel for D/A conversion, and the gray shade codes appear serially.

The output shifting function of the Raster Line Composition Unit is Multiplexed four to one, so that the desired 40 MHz element rate can be achieved with internal shifting rates of 10 MHz.

The bulk of the equipment in the Video Assembler is contained in the two sets of storage elements. Each of these has as many locations as there are elements in a line and as many bits per location as there are bits in the gray shade code. The complexity of this part of the system can thus be seen to be roughly proportional to both the number of elements and the number of bits in the code. The number of environment edges, while not contributing so obviously to Video Assembler complexity, has a profound effect on the design because it determines the face loading rate. With 500 environment edges, the time available for face loading is sufficient that no real difficulty occurs. About 1000 environment edges could be accommodated by refinements of the same approach to loading design; with such a number of edges, the Queue would definitely be necessary.

Eventually, the designer comes upon a fundamental limitation on the number of faces that must be loaded during a raster line. Namely, there can never be more faces appearing on a line than there are elements in that line. This occurs because the faces are loaded in priority order, and the least extent that any face can have on a raster line is one element. There may be more faces intersecting the raster line than can be seen, but if time runs out and these invisible faces cannot be loaded, nothing is lost. In fact, it appears that the likelihood that, say, 1000 edges would be used by the environment designer to make 333 triangles, and that all 333 of these triangles would be visible on a single raster line, must be exceedingly small. But how small is this likelihood and how bad will the result appear if it does occur? More is said on this subject in paragraph C.

4. D/A Converter Unit

The D/A Converter Unit performs two functions; it demultiplexes the four input data lines into a single 40 MHz video line, and it converts the six-bit gray shade codes to analog voltages for application to the display CRT. This design will be implemented with MECL II circuits. The conversion from TTL logic levels to those of MECL II is done before the demultiplexing. The D/A conversion is accomplished by high-speed current mode conversion circuits similar to those that have been used for the same task in previous visual simulators.

5. Conversion to Color Operation

In order to have full color operation, assuming that a suitable display is available, a color memory would be interposed between the Raster Line Composition Unit and the D/A Converter as shown in Figure 37. The six-bit codes would now become addresses and would specify any one of 64 locations in the color memory, each of which would contain the digital representation of the red, green, and blue components of the desired color to be displayed. Each of these three color components would be applied to a D/A converter, just as the gray shade code is for black and white operation, and the three results would be applied as the red, green and blue drive to the display.

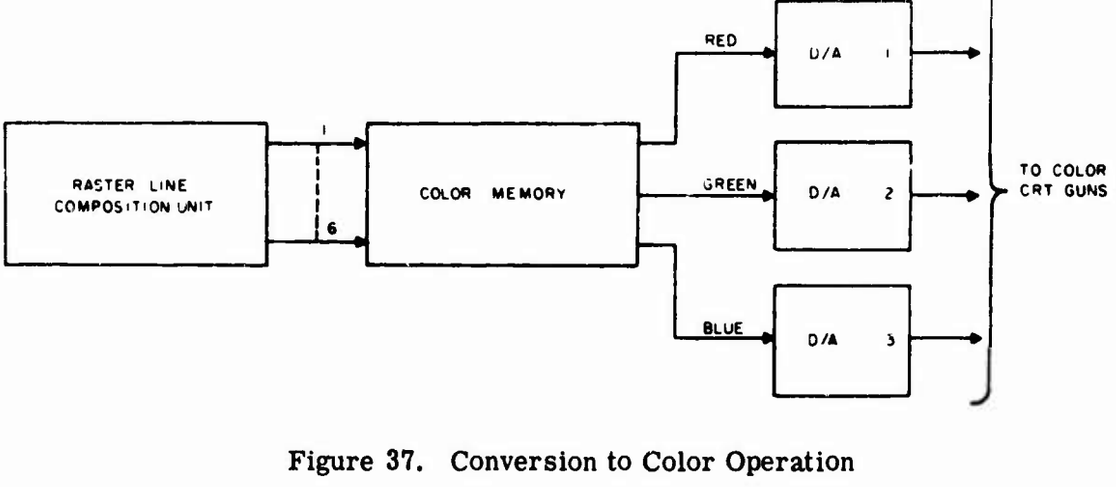


Figure 37. Conversion to Color Operation

The number of different colors used in an environment would depend on the number of bits in the code. For the six-bits currently discussed, for example, 64 different colors could be stored in the memory. The actual composition of the colors in the memory, however, depends on the length of the words stored there. Suppose, for example that the red, green and blue components are each specified by five bits. Then the overall color word stored would have a length of 15 bits, and the total number of possible different colors that could be stored would be $2^{15} = 16,384$.

The color memory itself should probably be multiplicated to limit the access time demands that would be placed on it. Use of a separate memory for each of the four multiplexed levels of the shifting Raster Line Composition Unit would allow 100 nsec for each access to achieve the 40 MHz video rate desired. Such a memory can be constructed of commercially available bipolar MSI memory arrays. Since the color memory is not very large (perhaps 64 words of 15 bits each), such multiplication is not extremely expensive.

C. AREAS FOR FURTHER INVESTIGATION

Three profitable areas for further investigation are the gathering of scene statistics, further investigation of the Texas Instruments LSI program, and the continuation of semiconductor industry evaluation.

In the area of scene statistics, it is important to determine the factors that specify the length of the loading queue and the upper limit on the number of faces that must be loaded during a raster line period. This information can be determined from statistical data gathered from actual scenes that have different numbers of edges and different object content. What must be determined is the fraction of time that the number of visible faces per line or the face determination rate exceeds a certain value, as a function of that value.

Considerable information now exists on the complexity that can be achieved with the Texas Instruments LSI approach. The confidence that the arrays specified for the proposed system framework can be built is quite high. The next step that should be taken involves conferring with Texas Instruments on the implementation of specific logic functions, in order that the options in

logic design are chosen such that the LSI implementation of the function is simplified. Then, finished logic diagrams should be sent to Texas Instruments for consideration, so that the ease or difficulty of the required implementaton can be determined.

The survey of semiconductor industry capabilities should be continued. This will be necessary in order to assess the progress of the expected technological advances - particularly in MOS - and to gather new information about standard MSI products that are available.

IX. SURFACE GENERATING SUBSYSTEM

A. INTRODUCTION

When evaluating approaches for the generation of environments using digital computer techniques, certain terrain characteristics must be considered. In general, two distinct types of terrain must be generated: large areas with essentially random texture such as fields, lakes, and forests, and small areas with much unique texture such as villages and urban areas. It is desirable to maintain the amount of detail at a high level and retain the visual cues. However, the ability to assign a unique color to every point is not necessary, or even desirable from the user's standpoint (a twenty-mile square contains 10^{10} points one foot apart). The eventual technique implemented should be capable of generating these types of terrain and should, if possible, be constructed to take advantage of these characteristics.

One technique that holds much promise is the Scanner and Map Unit technique described in the following paragraph. The Scanner generates the coordinates of the raster vector projected into the ground, and the Map Unit generates video data from the coordinates. The advantages of this approach are the economical generation of detail with a limited amount of input data and the capability to perform smooth resolution changes (causing detail to fade as it approaches the raster structure in size and thus avoiding scintillation). The disadvantages of this approach are that rectilinear textures are generated and that each scanner is tied to a coordinate system; thus two textured surfaces at angles to each other require two scanners.

B. SURFACE SCANNER

1. General Theory

The basic problem to be solved is the perspective transformation, which describes the raster as projected onto the ground plane. Given the raster vector components P_X, P_Y, P_Z and station point vector X_N, Y_N, H^* in ground coordinates, the equations to be implemented are

$$X = X_N + H^* \frac{P_X}{P_Z} \tag{20}$$

$$X = Y_N + H^* \frac{P_X}{P_Z} \tag{21}$$

Since the numerators and denominators are, in general, functions of the line and element numbers, the quotients change at the element rate. A divider with this speed requirement is not feasible with the digital logic presently available. One approach to circumvent this problem was used on the NASA system.

2. NASA Surface Generator Approach

The division problem was avoided in the NASA I system by rotating the raster structure on the CRT so as to make the denominators of Equations (20) and (21) independent of the element number. For example, for Equation (20):

$$X = X_N + H^* \frac{P_X}{P_Z}$$

Expanding P_X and P_Z in terms of line and element numbers,

$$X(i,j) = \frac{X_N + H^* \left[PCX + i\ PLX + j\ PEX\right]}{PCZ + i\ PLZ + j\ PEZ} \qquad (22)$$

By forcing the raster parallel to the horizon, PEZ = 0, and the equation becomes:

$$X(i,j) = X_N + \frac{H^*\ PCX + iH^*PLX}{PCZ + i\ PLZ} + j\ \frac{H^*PEX}{PCZ + i\ PLZ} \qquad (23)$$

The computational task for each coordinate has been reduced to two divisions every line and element-by-element additions. The raster is rolled electronically by performing a coordinate rotation on the horizontal and vertical sweeps in the NASA system.

There are a number of disadvantages to this approach. The CRT electronics become very complex with severe sweep circuit bandwidth requirements. The construction of mosaics of displays is not practical because of the edge matching problems. The use of large-screen displays such as light valves is also not feasible, since a projector with rolled raster has not been developed. To avoid these problems, a new technique has been developed.

3. Digital Scanner Concept

The Digital Scanner provides an approximation to X and Y which is always correct to the resolvable neighborhood of the projection of the raster into the ground. One feature of the current map technique is that patterns are nested so that detail can be added and subtracted according to the resolution of the ground. This is accomplished by keeping the information content essentially constant. Patterns are discarded when their image size becomes comparable to the raster element size. The set of texture pattern maps that are addressed contains basic pattern cells that are always larger than the display elements.

Consider computation of the X coordinate and assume that the pattern displayed contains cells of size S. To display cells of this size, it is sufficient to determine the cell that contains X, i.e., solve for an X_1 that specifies the memory address of the cell such that $X_1 \leq X \leq X_1 + S$. X_1 is then applied to the maps.

The exact solution for X is given by Equation (20). If, for some P(i, j), we know the X coordinate of the cell S that contains the image of P(i, j), then the image of any neighboring display element either lies in the same cell or in an adjacent one, because of the criterion for selecting S. Also since P(i, j) is monotonic, the corresponding X will change in the same direction as the entire line is passed. More concretely, let X_1 be the cell coordinate corresponding to the zeroth element of some line, i. Let j increase. Then, if X increases from j = 0 to j = 1, it will continue to increase across the display line. We need only check for entry into the next cell for $X \geq X_1 + S$. When this occurs, the X map coordinate is changed to $X_1 + S$. We monitor the condition:

$$X \geq X_1 + S$$

or

$$X_N + H^* \frac{P_X(i, j)}{P_Z(i, j)} \geq X_1 + S \tag{24}$$

X_1 may be rewritten as $X_0 + X_N$ where $X_0 = X_1 - X_N$. Then the equation may be rewritten as:

$$X_N + H^* \frac{P_X(i, j)}{P_Z(i, j)} \geq X_0 + X_N + S \tag{25}$$

or

$$H^* P_X(i, j) - P_Z(i, j) \left[X_0 + S\right] \geq 0$$

Breaking P_X and P_Z down into their components the equation becomes:

$$H^* \left[PCX + iPLX + jPEX\right] - \left[PCZ + iPLZ + jPEZ\right] \left[X_0 + S\right] \geq 0 \tag{26}$$

By inspection, the operation to be performed is the addition of

$$H^* PEX - PEZ(X_0 + S) \tag{27}$$

at the element rate into an accumulating quantity. When the inequality is satisfied (sign change or zero), the map coordinate is changed from X_1 to $X_1 + S$ and a new inequality is formed.

$$X^* P_X(i, j) - P_Z(i, j) \left[X_0 + 2S\right] \geq 0 \tag{28}$$

The working cell size must be adjusted so that it continues to satisfy its stated definition. This can be readily accomplished by monitoring the number of element additions required to satisfy the inequality. This number would be compared with a priori limits that regulate S. The general form of the inequality then becomes

$$H^*P_x(i,j) - P_Z(i,j)\left[X_0 + \sum_k m_k S_k\right] \geq 0 \text{ for X increasing}$$

$$H^*P_x(i,j) - P_Z(i,j)\left[X_0 - \sum_k m_k S_k\right] \leq 0 \text{ for X decreasing} \qquad (29)$$

where m_k is the number of cells of size S_k, which are added or subtracted according to the direction in which X is changing. Since the maps are addressed by binary numbers, it is appropriate to make the S_k integer powers of two, which simplifies the necessary multiplication to a simple shift.

In effect, the scanner solves the original division problem with an accuracy that varies to suit the resolution needs on a point-to-point basis. It does this by correcting the appropriate quotient bits, one at a time, with a maximum rate of one bit per element time.

4. Scanner Organizational Concepts

The scanner computation breaks down into the following operations:

1) Calculation of quantities unchanged for the entire frame.
2) Calculation of interlace quantities.
3) Calculation of starting point quantities for each line.
4) Element-by-element additions.
5) Update operations occuring when a cell boundary is crossed.

The scanner has been partitioned on the basis of the computation speed required and the type of computation to be performed. Items 1, 2, and 3, requiring full arithmetic capability at the horizontal line rate (or slower), are performed in the Scanner Arithmetic Section. Items 4 and 5, which require additions at the element rate, are performed in the X and Y Boundary Trackers.

5. Scanner Arithmetic Section

a. General

Figure 38 illustrates the operations in the Arithmetic Section. During vertical blank time, the Arithmetic Section accepts inputs from the system G/P computer, percomputes the quantities that are constant during the frame, and computes the interlace quantities. During the active horizontal line time, the Airthmetic Section computes the X and Y Boundary Tracker starting point quantities. During horizontal blank time, the Arithmetic Section transfers the starting data to the Trackers.

b. Detailed Tasks

(1) Vertical Blank Time

(a) Accepts P and L vectors (12 words) from system G/P computer.

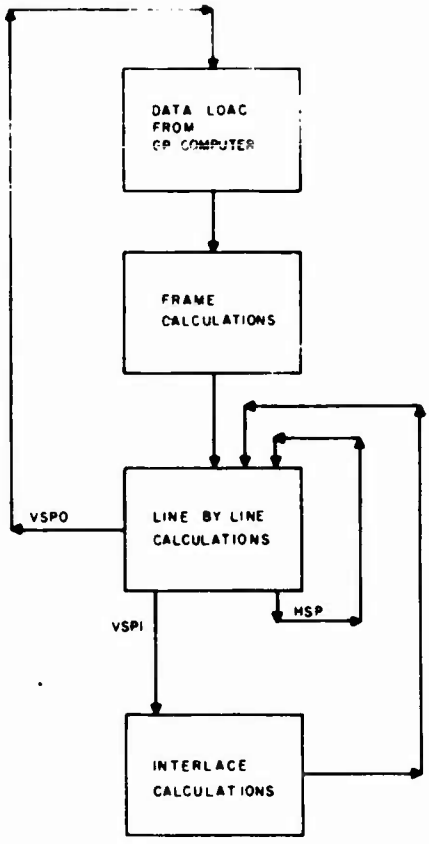


Figure 38. Computation Algorithm Surface Scanner - Arithmetic Unit

(b) Precomputes H*PCX, H*PLX, H*PEX, P*PCY, H*PLY, H*PEY, PEX/PEZ, PEY/PEZ.

(c) Computes XY_{BOR}, S_{BOR}.

The use of these quantities is explained in the following section on Arithmetic Section computation during the active horizontal line time. This particular calculation has not yet been completely defined but it does appear feasible to compute these quantities once per frame. Further investigation is required.

(d) Computes interlace quantities during odd field vertical blank time.

$$PCX + 1/2\ PLX = PCX'$$
$$PCY + 1/2\ PLY = PCY'$$
$$PCZ + 1/2\ PLZ = PCZ' \qquad (30)$$

(e) Transfers PEZ to Boundary Trackers.

(2) Active Horizontal Line Time

During the active horizontal line time, the Arithmetic Unit calculates the Tracker starting quantities for the next line. These data are computed in two ways, depending on the position of the raster vector at the start of the line. In the discussion of the Scanner Concept (Section IX. B. 3), the assumption was made that the cell size was equal to two elements or more. This assumption is correct for scanner operation in the regions where textured surfaces are to be drawn, but is incorrect, ingeneral, for the starting point of a raster line. If the raster vector projects into the sky or into the region near the horizon where the cells are essentially infinite in size, the scanner cannot draw resolvable cells. However, it is possible for the vector to scan into a region where resolvable cells should be drawn. The solution to this problem is to compute the border of the region of resolvable cells and use this value as the first boundary to be crossed. Then, if the Tracker crosses this boundary, cell generation can proceed in the normal manner. From the studies conducted thus far, it appears feasible to compute the border value and the appropriate cell size once per frame and use them to compute the Tracker starting values when required. This area has not been thoroughly investigated and more study is required. Proceeding under the assumption that this approach will work, the decision structure in Figure 39 results.

The first task to perform is the generation of P_X, P_Y, and P_Z for the next line.

$$P_X(i, o) + PLX = P_X(i + 1, o)$$

$$P_Y(i, o) + PLY = P_Y(i + 1, o)$$

$$P_Z(i, o) + PLZ = P_Z(i + 1, o) \qquad (31)$$

The first decision to be made is to determine whether the raster vector is in the ground or in the sky. If the sign of P_Z is negative, the vector is in the sky and both sets of Tracker inputs must be computed using the border value. The next decision to be made is which border will be crossed, the positive border or the negative border. This decision must be made independently for both Trackers and is made by comparing PEX/PEZ TO PX/PZ for the X Tracker and PEY/PEZ to PY/PZ for the Y Tracker. If PEX/PEZ (or PEY/PEZ) is greater than P_X/P_Z (or P_Y/P_Z), then the positive border value should be used. If less, then the negative border value should be used.

If the sign of P_Z is positive, the raster vector is in the ground and the starting point is computed using the equations

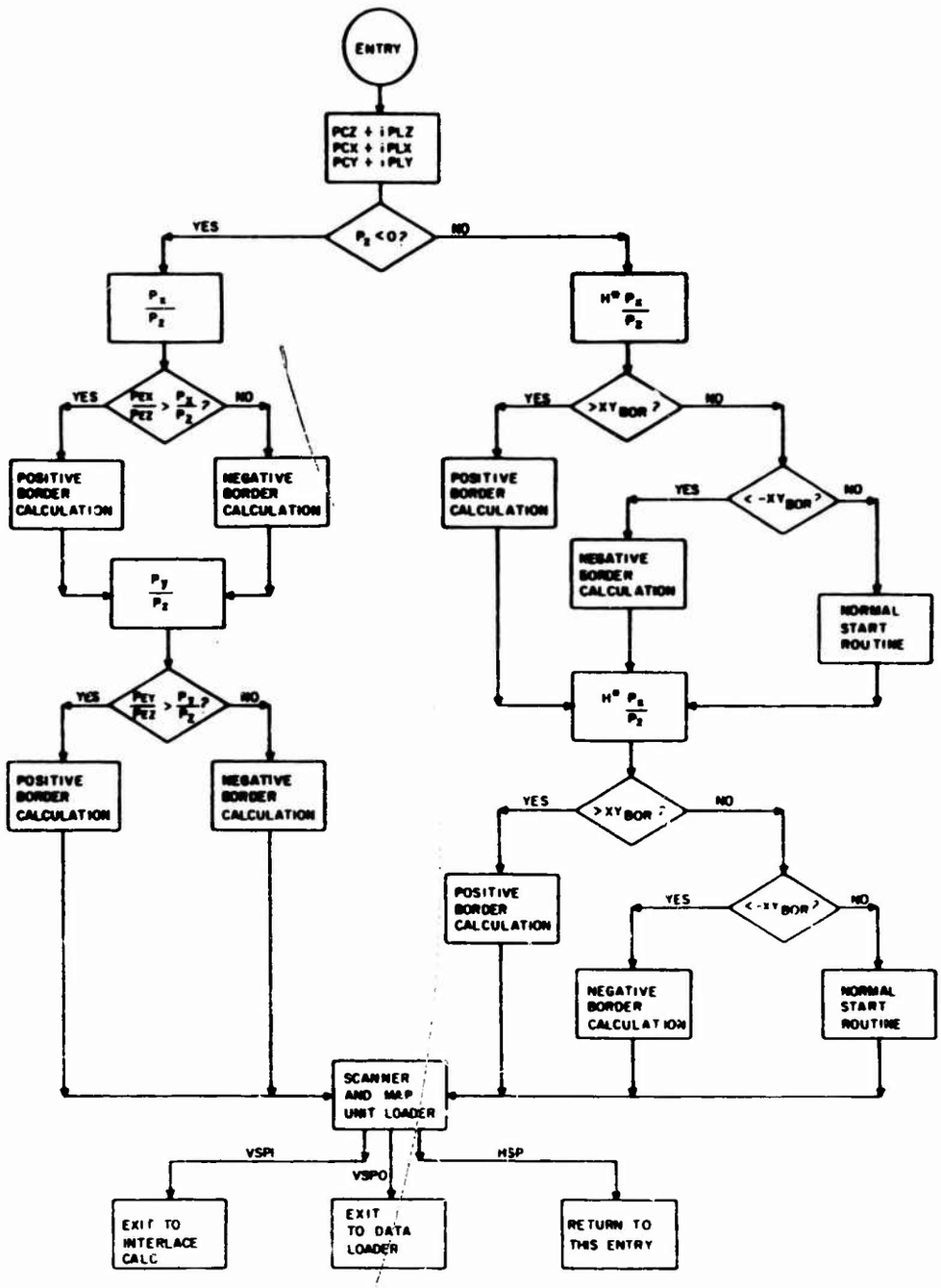


Figure 39. Line-by-line Algorithm Scanner Arithmetic Section

$$X = \frac{H^*P_X(i,0)}{P_Z(i,0)} \quad \text{and} \quad Y = \frac{H^*P_Y(i,0)}{P_Z(i,0)} \qquad (32)$$

The next decision is to determine whether the starting point is outside of the resolvable region. This decision is made by comparing the magnitudes of the starting point values to the border value. As before, this decision must be made independently for each Tracker. If the computed starting value is positive and its magnitude is greater than the border value, then the border value should be used as the initial value for the Tracker. If the computed starting value is negative and its magnitude is greater than the border value, then the negative of the border value should be used as the initial value for the Tracker. The actual calculation of initial Tracker values using the border values has not been discussed, since it is similar to the calculation on initial Tracker values using computed starting values, as discussed in the following section.

In either case, if the raster vector is in the sky or outside of the resolvable region of either or both Trackers, the appropriate saturation indicators should be set for transfer to the Trackers during horizontal blank time. These indicators are used in the Map Unit to indicate when to draw background colors.

If, however, the sign of P_Z is positive and the magnitude of the computed starting value is less than the border value, then the "normal" routine can be followed for the calculation of initial Tracker values. The routine is performed in four steps. First, the cell size is found. Then the starting map coordinate is computed. Next, the first boundary crossing is computed. Finally, the initial Tracker values are generated.

The cell size is generated by computing the coordinates of a point three elements away from the computed starting point and subtracting these coordinates from the coordinates of the computed starting point. This calculation also generates the quantities "sign delta x" and "sign delta y" used in the Trackers to indicate the direction of the update operation - positive or negative:

$$S_X = X - \frac{H^*\left[P_X(i,0) + 3PEX\right]}{P_Z(i,0) + 3PEZ} \qquad S_Y = Y - \frac{H^*\left[P_Y(i,0) + 3\,PEY\right]}{P_Z(i,0) + 3\,PEZ} \qquad (33)$$

A point three elements away was chosen, because the following rule was set up for cell size modification: "If two elements per cell, increase the cell size. If three elements per cell, maintain the present cell size. If four elements per cell, decrease the cell size." This rule has been adequate in the

simple cases studies thus far, but more study is necessary to verify that it will work in the general case.

The starting map coordinate is generated by adding the computed starting value to the nadir coordinate of the station point and rounding the sum down at a point indicated by the cell size.

$$X_0' = \left[X + X_N\right]_{\substack{\text{Rounded down} \\ \text{by } S_X}} \qquad Y_0' = \left[Y + Y_N\right]_{\substack{\text{Rounded} \\ \text{down by } S_Y}} \tag{34}$$

The sum of the computed starting point and the station point coordinate is then rounded up at the same point. The station point coordinate is then subtracted from this quantity to get the first boundary crossing

$$x_o' = \left[X + X_N\right]_{\substack{\text{Rounded up} \\ \text{by } S_X}} - X_N \qquad y_o' = \left[Y + Y_N\right]_{\substack{\text{Rounded up} \\ \text{by } S_Y}} - Y_N \tag{35}$$

The first boundary crossing is then used to compute the Tracker input quantities, as follows:

$$\begin{aligned} \Sigma_x &= H^* P_x(i,o) - x_o' P_z(i,o) \\ E_x &= H^* PEX - x_o' PEZ \\ \Sigma_y &= H^* P_y(i,o) - y_o' P_z(i,o) \\ E_y &= H^* PEY - y_o' PEZ \end{aligned} \tag{36}$$

(3) Horizontal Blank Time

During this time, the Arithmetic unit must transfer the data to the boundary trackers. The transfer is as follows:

To the X boundary tracker:

(a) $\Sigma_x = H^* P_x(i,o) - x_o' P_z(i,o)$

(b) $E_x = H^* PEX - x_o' PEZ$

(c) $P_z(i,o)$

(d) X_o'

(e) S_x

(f) Sign delta x

(g) X saturation indicator

To the Y boundary tracker

(a) $\Sigma_y = H^* P_y(i, o) - y_o' P_z(i, o)$

(b) $E_y = H^* PEY - y_o' PEZ$

(c) $P_z(i, o)$

(d) Y_o'

(e) S_y

(f) Sign delta y

(g) Y saturation indicator

c. Summary of Computational Tasks in Scanner Arithmetic Unit

(1) Vertical Blank Time

Even field - (a) 12 data word transfer from system G/P computer

(b) 6 multiplications

(c) Calculation of border values and cell sizes (not completely defined at present)

(d) 1 word transfer to both Trackers

Odd field - (a) 3 additions

(2) Active Horizontal Line Time

The following summary assumes the Tracker data is computed using the "normal" routine. Calculations using the border values are essentially identical.

(a) 20 additions/subtractions
(b) 4 divisions
(c) 4 multiplications
(d) 6 program jumps on data
(e) 6 shift/roundoff operations
(f) 1 halt for horizontal sync pulse

(3) Horizontal Blank Time

(a) 7 data transfers to X Boundary Tracker
(b) 7 data transfers to Y Boundary Tracker

d. Hardware Requirements for Scanner Arithmetic Unit

The design of the Scanner Arithmetic Unit is essentially dictated by the computational tasks to be performed during the active horizontal line time. Analysis of the types of operations indicate that stored program capability is desired. The speed requirements indicate a very high-speed arithmetic section. The eventual design will be very similar to that of a small general-purpose computer with an input/output section, a memory for data and program storage, an instruction decoding and execution section and a very-high-speed arithmetic section.

6. Scanner Boundary Trackers

a. General

The X and Y Boundary Trackers accept inputs from the Scanner Arithmetic Section during horizontal blank time and generate X and Y coordinates, cell sizes, and control signals for the Map Unit during the active horizontal line time. Each Tracker consists of an arithmetic section which accumulates terms in the equation

$$H^{*}P_x(i,j) - P_z(i,j)\left[x_o' \pm \sum_k m_k S_k\right] \qquad (37)$$

as i, j, and $\sum_k m_k S_k$ vary, and a control section which monitors the sign of the result, determines S, and controls element and update additions. The arithmetic section is further partitioned into five blocks: the accumulator block containing the accumulator register and associated element/update shifting adder, the element addend block containing the element addend register and associated update shifting adder, the Z vector block containing the P_Z and PEZ registers and associated element adder, and the map coordinate block containing the map coordinate and cell size registers. See Figure 40. Since the X and Y Trackers are identical, all blocks are duplicated for a complete Scanner except for the Z vector block which can be shared. The following discussion on Tracker operation is couched in terms of the X Tracker but all comments also refer to the Y Tracker.

b. Tracker Operation

The X Tracker operates in three modes: Data load (during horizontal blank time), element addition (between cell boundaries during the active horizontal line time), and update addition (when a boundary crossing has been detected).

During the data load mode, the following data transfers occur:

1) Σ_x is loaded into the accumulator register.

2) E_x is loaded into the element addend register.

3) $P_z(i,o)$ is loaded into the P_Z register.

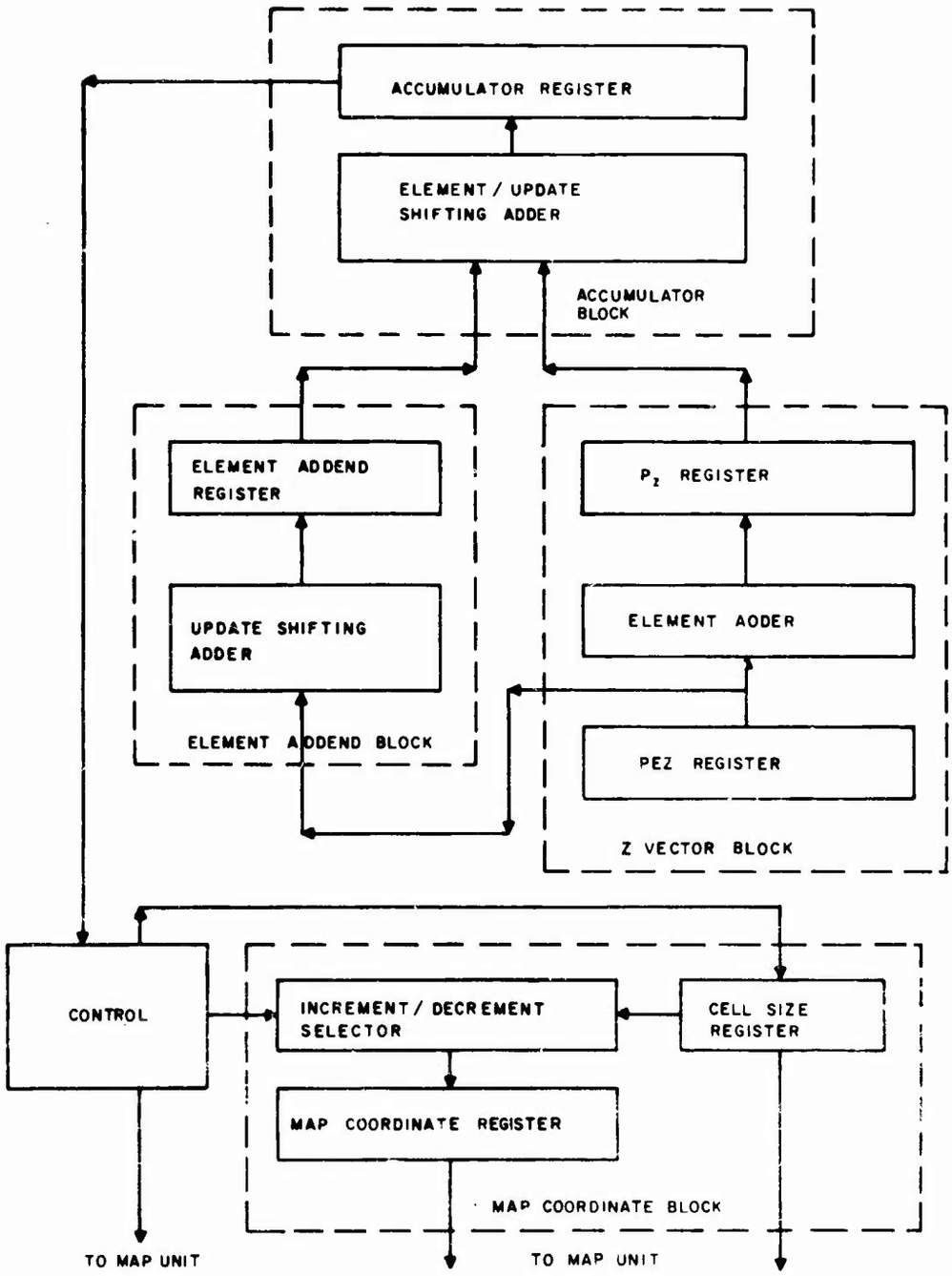


Figure 40. Boundary Tracker Partitioning

4) X_o' is loaded into the map coordinate register.

5) S_x is loaded into map cell size register and control storage.

6) Sign delta x is loaded into control storage.

7) X saturation indicator is loaded into control storage.

In order to minimize the date transfers occurring during horizontal blank time, PEZ is loaded into the PEZ register during vertical blank time.

During element addition, the contents of the element addend register are added to the contents of the accumulator register through the accumulator adder. PEZ from the PEZ register is added to the contents of the P_Z register through the Z vector adder. The cell size monitor in the control section is monitoring the number of element additions per boundary crossing to determine if the cell size should be changed.

The update addition is initiated when a sign change or zero condition occurs in the accumulator register and P_Z is positive. The element addition is disabled and the saturation indicator is reset if it was set. P_Z, scaled by the cell size, is added to the contents of the accumulator register. The scaling is performed in the accumulator adder. PEZ, scaled by the cell size, is added to the contents of the element addend register. The scaling is performed in the element addend adder. The sign of P_Z and PEZ is determined by the sign delta x quantity from the control section and P_Z and PEZ with the proper sign are generated in the Z vector block. The map coordinate register is incremented or decremented depending on the sign delta x quantity. The position in the map coordinate register at which the increment or decrement operation is performed is determined by the cell size. PEZ from the PEZ register is added to the contents of P_Z register through the Z adder vector. If the cell size monitor indicates that the maximum cell has been reached and a further increase is required, then the saturation indicator is set. During the first element addition after an update operation, twice the contents of the element addend register are added to the contents of the accumulator register to restore the correct relationship between the raster and the accumulator register.

The output to the Map Unit from a complete Scanner consists of the X and Y coordinates, the X and Y cell sizes, the X and Y saturation indicators and the sign of P_Z.

C. SURFACE MAP UNIT

1. General

The Map Unit generates textured surface video from the X and Y coordinates, cell sizes and control signals generated in the X and Y Boundary Trackers. The Map Unit has four requirements:

1) It must be capable of generating the two types of terrain previously discussed; large areas with random detail and small areas with unique detail.

2) It must be capable of performing smooth resolution changes and fading detail as the size of the detail approaches the raster cell structure.

3) The user should be able to modify the texture patterns.

4) The displayed patterns should be multi-colored.

2. Approach

The recommended technique which fulfills the above requirements is an expanded version of the technique used on the NASA system. In this system, texture information is stored in a set of maps and the X and Y coordinates of the projected raster vector are used to address the maps. The maps are simple forms of storage providing one-bit output to designate one of two possible colors. Complex networks of patterns are obtained by logically combining the outputs of a number of maps each of which contributes to the textural detail over repetitive regions of a particular size. The regions are structured so that the patterns are nested within each other. This hierarchy of patterns makes it possible to transition from one level of detail to another by deleting the contribution from the map whose cells approach the raster cell structure.

The expansion of this technique to fulfill the requirements discussed above involves the use of recent advances in high-speed logic and memories. The texture patterns are stored in read/write memories allowing for ease in pattern modification. All levels other than the highest contain multiple maps. Each cell of the maps contains a color number (used for selecting color data from a color memory) and a selection code to identify the map used for generating finer detail. The X and Y coordinates from the Boundary Trackers identify the cell of the maps of a particular level to be interrogated and the selection code from the higher level map determines the map output to be used. The color number from each level is applied to a color memory to extract the color data. The color data from two levels of maps are combined in a weighted sum to form the final color output. The selection of levels and the weighting of the sum are determined by the larger of the two cell sizes from the Boundary Trackers.

3. Hardware Considerations

The Map Unit is partitioned into six blocks; the input processing block, the four texture blocks, and the output processing block as shown in Figure 41.

The input processing block accepts X and Y coordinate data, cell sizes, and control signals from the Boundary Trackers. It generates addresses for the texture blocks from the X and Y coordinates, selects the larger of the two cell sizes for use in the output processing block, and transfers the two saturation indicators and the sign of P_z to the output processing block.

The third order texture block contains seven texture pattern maps. This block accepts addresses from the input processing block and a selection code from the fourth order texture block. It generates a color number for use in the output processing block and a selection code for use in the second order texture block.

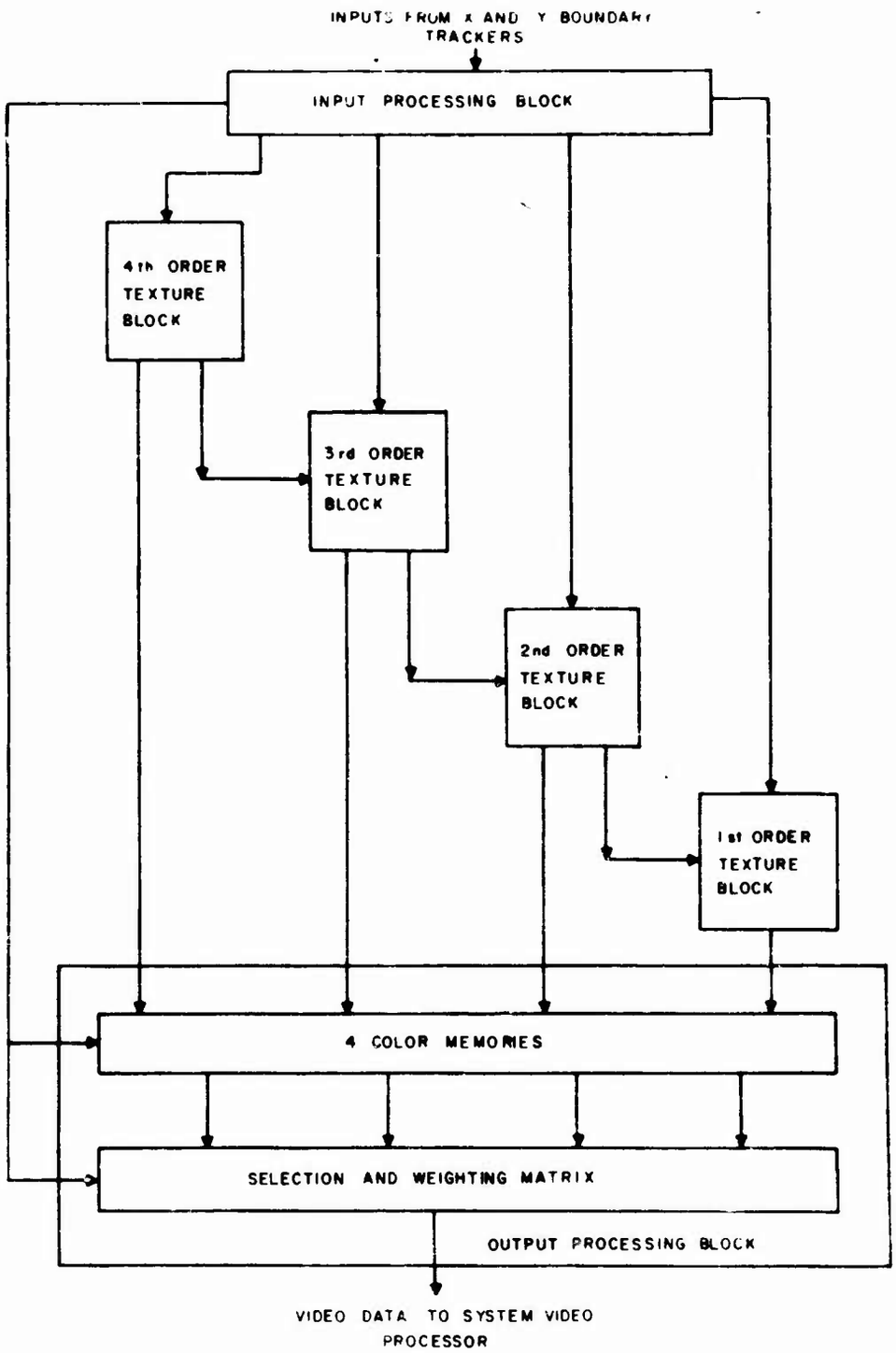


Figure 41. Map Unit Partitioning

The second order texture block contains seven texture pattern maps. This block accepts addresses from the input processing block and a selection code from the third order texture block. It generates a selection code for the first order texture block and a color number for use in the output processing block.

The first order texture block contains seven texture pattern maps. This block accepts addresses from the input processing block and a selection code from the second order texture block. It generates a color number for use in the output processing block.

The output processing block generates video data from the color numbers from the four texture blocks and the cell size, saturation indicators, and sign of P_Z from the input processing block. This block contains four color memories and the selection and weighting matrix used for generating video data. If the sign of P_z is negative, the output processing block generates the sky color. If either saturation indicator is ON and the sign of P_Z is positive, the output processing block generates the horizon color. If the sign of P_Z is positive and the saturation indicators are OFF, the output processing block generates a video output which is the weighted sum of two color memory outputs where the selection and weighting is determined by the cell size.

D. SUMMARY

During this study, the effort in this area was directed toward developing the scanner concept to the point where reasonable estimates can be made of the hardware complexity. There are a number of areas in which further investigation is required, however, before the concept could be implemented. First, the scanner algorithm should be simulated on a computer with special attention paid to the border calculations previously discussed. The ability of the scanner to track boundaries along a raster line should also be checked to determine the validity of the cell size modification rule discussed. Next, a scaling and error analysis should be performed in order to determine word lengths at various points in the scanner operation. This would allow partitioning and logic speed problems to be considered. Finally, the Map Unit should be simulated on a display where the experimenter can observe the results as texture pattern video. Most of the important characteristics of maps can not be evaluated without looking at their images under dynamic conditions.

X. POINT SOURCE GENERATION

A. GENERAL

The principal task of the point source generator is to provide the large number of ground-level lights encountered in night flying in the vicinity of an airport. In the discussion of training tasks in Section II, it was estimated that approximately 500 light sources would be needed to adequately depict lighting for one runway and some taxiways. Object techniques must be ruled out immediately because it would take three edges to form even a triangular light source, a minimal representation.

Most of the lights associated with an airport are ordered and evenly spaced. The runway and taxiway lights account for 400 of the estimate requirement for 500 lights and they consist, for the most part, of parallel strings. Because of this regularity it seemed worthwhile to see if the surface generating technique could not be adapted to this task. If it were practical, then the point source generator could be eliminated or at least drastically reduced in size.

Two difficulties are encountered in attempting to use surface techniques for this application. First, the map structure would limit economical use to light strings oriented parallel to the X and Y ground axes and require that the spacing between lights be an integral power of two. Although this might be somewhat inconvenient, it does not rule out this approach. The second problem is associated with resolution. The surface technique projects the raster P vector onto the ground and the (X, Y) intersection point is found. The intersection found for each display element corresponds to the center of that element. The ground surface is therefore sampled at these points (which may be widely spaced relative to the size of the lights) making it possible to miss some of the lights. What are really desired are the ground coordinates corresponding to the four corners of each display element. If these were available, then one would merely check to see if the maps contained a light anywhere in this quadralateral area on the ground.

Several computational schemes were investigated to solve the problem in this way. Part of the desired information is computed in the surface scanner discussed in Section IX. The cell size, S, gives an adequate indication of the difference in ground coordinates from element to element along a raster line. The associated differences from line to line are not available and the difficulty of computing them caused this approach to be abandoned.

A separate point source generator was therefore found necessary. The point source images are found by direct computation of the display line and element numbers as discussed in Section III.A.3. The computation is performed in the General Purpose Computing Subsystem and occupies one of the two central processors for almost an entire frame. Although this appears to

be the most economical method for the present requirement, it is not readily expandable. The computation of a large number of points would require a small special purpose arithmetic unit. Its primary tasks are addition and division and its design could utilize appropriate sections of the surface generator arithmetic unit.

B. POINT SORTING AND STORAGE

The point source images are computed during the frame in the order in which they appear in strings of lights in order to simplify the computations. The image points must then be sorted by line and element number so that they are in proper order for raster display.

Data are transmitted to the sorting unit as they are computed and each is placed in its proper position in the list while the next value is computed. By the end of a frame, a completely sorted list is formed for the following frame. Two sorting and storage units are needed so that one is available for readout while the other one is being loaded.

Advantage is taken of the relative ease with which view assignment may be made for points. The sorting and storage unit is designed to accommodate 500 points and these may be assigned as necessary between the two views. This is an economical approach to operating two views with a common station point in a 500 point environment. It does not allow the two views to operate independently where each is in a maximum point source environment.

The data word from the General Purpose Computing Subsystem contains 27 bits for each point and has the format shown in Figure 42. The display plane position is given by 10 bit line and element numbers. A one bit tag designates the view to which the point is assigned. A full six-bit color code has been included for maximum flexibility although two bits would suffice for the colors usually needed. The smaller number of bits was not used because the savings would be slight and the larger number of bits would be needed for black and white operation in any case.

10 Bit Line No.	10 Bit Element No.	1 Bit View	6 Bit Color Code

Figure 42. Point Word Format

The point sorting and storage unit is basically a 500 stage shift register as shown in Figure 43. Each stage has 27 elements. For purposes of sorting, the 21 bits comprising the line and element numbers and the view assignment bit are regarded as the order number. The view assignment bit is included in the ordering number so that it is possible to have points fall on identical elements of both displays. The sorting process places the first point to be encountered during the raster scan in the first stage, and so on, by the following process.

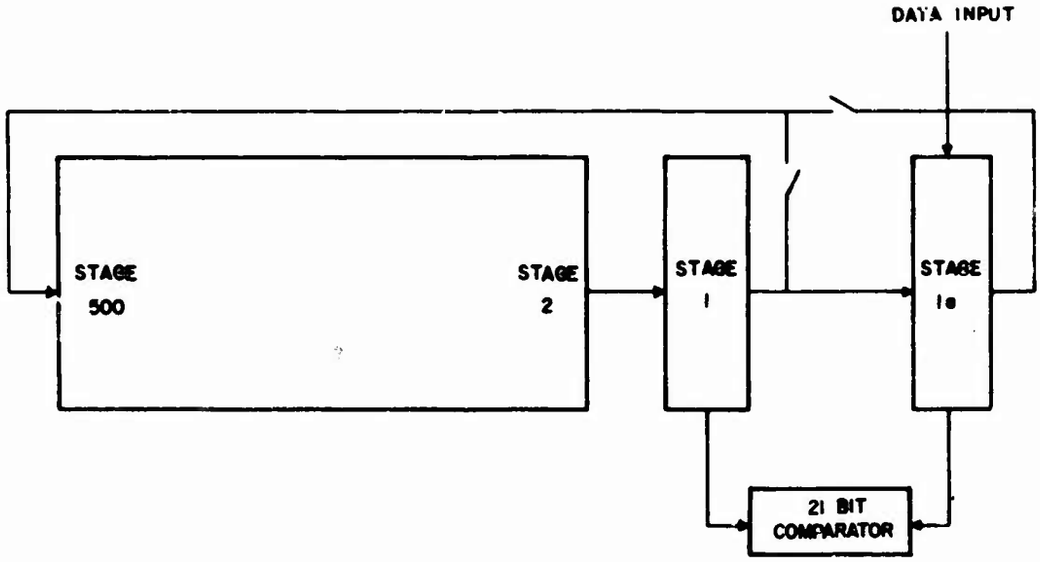


Figure 43. Point Sorting Technique

At the start of the frame, the 21-bit sorting numbers are all set to their maximum values and the color entries are cleared to zero. From this point on, each new data word is deposited in temporary storage stage 1a. The entire register is circulated with stage one connected to stage 500 and the path from stage 1a open while a comparator checks the entries in stages 1 and 1a. When stage 1 is found to be greater than stage 1a, the circulation loop is modified. The connection between stage 1 and stage 500 is opened and the one between stage 1a and stage 500 is closed. This inserts the new data in its proper position. If the comparator detects an "equal" condition on all 21 bits, the new entry is discarded as it would fall on top of a previous entry. In all cases, the 500 stage register is shifted 500 times for each new entry.

Approximately 25 milliseconds are available for computing points. Therefore, new data are supplied each 50 microseconds which implies that the shifting rate of the register is approximately 10 MHz during the sort operation.

The shifting rate during the video generation time is a function of the arrangement of the points relative to the raster. During read-out, a line and element comparator would monitor the scanning process and indicate the occurrence of a match between the contents of stage one and the present position of the raster scan. When match is detected, the color code would be gated to the proper view. (It is assumed that both displays are scanned synchronously.) The register is then shifted right placing the next point in position. Since it is possible to have a number of points on adjacent elements, the maximum output shifting rate is governed by the video rate. For this reason, the actual implementation of the register shown in Figure 45 would use multiplexing. By using four registers of 128 stages each and operating them at a 10 MHz maximum shift rate, it is possible to accommodate points on adjacent elements in one view or points on every other element in two views.

A separate line and element output comparator would be required for each of the multiplexed registers.

The point source generator has been planned so that it can place points either on single lines or on line pairs. Point lights will probably require display on line pairs to avoid flicker unless viewing brightness is very low. A field correction is necessary in the output comparison if the points are to be displayed on the proper lines in both fields. Figure 44 snows two situations which require slightly different treatment. Two point images are shown in the interlaced raster structure and arrows show the proper display lines in each case. The 10-bit line number is computed by truncating fractional bits (not rounding). Thus any line number computed to be greater than or equal to zero, but less than one, would be truncated to zero. If the computed line number for the frame (10 bits) is even, then the nine most significant bits serve for both even and odd field read-out. If the computed number is odd (as for the lower point shown), the even field line comparison must be modified. The proper line is given by the most significant nine bits plus one least significant bit. The odd field is not affected. Therefore, the rule is: use the most significant nine bits of the line number on the odd field and the most significant nine bits of one plus the ten bit line number on the even field for line comparisons.

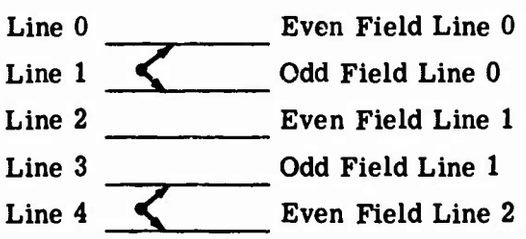


Figure 44. Field Correction for Two Line Display

XI. HARDWARE COMPLEXITY

A. GENERAL

The following information is supplied to indicate the approximate complexity of the various sections of the Image Generator. These estimates are based on the use of presently available components. In some instances, alternate methods of implementation would likely appear more attractive in the near future because of the rapid progress being made in medium and large scale arrays. However, we have chosen not to prejudge the outcome of these development efforts in considering these equipment estimates.

Because a number of different circuit packages are to be used, three different card formats have been assumed. Large format cards are necessary in several places to allow close grouping of certain logic functions. The approximate sizes of the card types would be: 10" × 5" (type I), 10" × 10" (type II), and 10" × 14" (type III).

No detailed design has been performed on any sections of the image generator and the numbers given are therefore only rough estimates. They include logic to control and perform the required functions. Component estimates do not include logic for test functions, spares, etc. Power supplies, backplane interconnection boards, cooling, and other cabinet related equipment are also omitted.

B. ESTIMATES

1. General Purpose Computing Subsystem

(2) Central Processors
(2) Teletypewriters
(2) Multiple access input/output processors
(1) Core memories, 8192 words
(1) Core memory, 16384 words
(2) Priority interrupt chassis (8 channels)
(1) Paper tape unit (read and punch)
(1) Random access disc and controller (optional)
(1) Card reader (optional)
(1) Line printer (optional)
(1) Magnetic tape unit (optional)

2. Object Calculating Section

(55) Type II cards (23 different designs)
(4600) MECL II integrated circuits
(800) Discrete components (transistor, etc.)

3. Object Processing Section

(56) Type II cards (22 different designs)
(32) Type III cards (identical)
(208) Texas Instrument large scale arrays (3 designs)
(3600) Standard medium scale arrays
(4200) MECL II integrated circuits

4. Surface Generating Subsystem

(92) Type I cards in scanner (25 different designs)
(6000) MECL II integrated circuits
(206) Type I cards in map unit (7 different designs)
(13400) MECL II integrated circuits

5. Point Sorting and Storage

(9) Type III cards (3 different designs)
(24) Texas Instrument large scale arrays (identical)
(250) Standard medium scale arrays
(550) MECL II integrated circuits

6. Video Processing

(20) Type II cards (4 designs)
(240) Standard medium scale arrays
(800) MECL II integrated circuits.

REFERENCES

1. Electronic Scene Generator installed at NASA Manned Spacecraft Center, Houston, Texas. Completed by General Electric 1967.

2. Warnock, J., "Hidden Line Algorithms and Halftone Picture Representation", (paper presented at Pertinent Concepts in Computer Graphics Conference, University of Illinois, Urbana, March 31, 1969).

BIBLIOGRAPHY

"Design News", EDN, April 15, 1969, pp. 24-26.

"Design News", EDN, May 1, 1969, pp. 18, 19.

"An LSI Approach... TTL Micromatrix Arrays", Fairchild Semicond., Mtn. View, Cal.

"Micromosaic Arrays...An MOS Approach to Custom LSI", Mtn, View, Cal.

"4500 Micromatrix Array Design Handbook", Mtn. View, Cal.

"4600/4700 TTL Micromatrix Array Design Handbook", Mtn. View, Cal.

"FairSim User's Manual", Mtn. View, Cal.

"Now It's MTNS", General Instrument. Electronic Design 9, April 26, 1969, pp. C38, C39.

Harary, Graph Theory and Theoretical Physics, Academic Press, N.Y.C., N.Y., 1967.

Harary, et. al., Structural Models: An Introduction to the Theory of Directed Graphs, J. Wiley, N.Y.C., N.Y., 1966.

Integrated Circuit Engineering Basic Technology, Fifth Edition. Integrated Circuit Engineering Corp., Phoenix, Arizona, 1966.

Bipolar/MOS Large Scale Integration. Integrated Circuit Engineering Corp., Phoenix, Arizona, 1968.

Mrazek, Dale. "Shrink Delay Line Costs with MOS", Electronic Design 5, March 1, 1969, pp. 50-57.

Speer, Raymond Daniel. "MOS on the Upswing", Electronic Design 8, April 12, 1969, pp. 49-79.

ECL 2500 Series Integrated Circuits, Texas Instruments, Inc., Dallas, Texas.

MOS Integrated Circuits, Texas Instruments, Inc., Dallas, Texas.

APPENDIX

MOS CONSTRUCTION, OPERATION, AND PERFORMANCE

A. CONSTRUCTION AND OPERATION

A simplified description of the method of construction and the principles of operation of metal-oxide-semiconductor (MOS) circuits is given below.

1. Basic Process

Four different types of MOS transistors can be made. Devices can be constructed with either N or P type material for the channel and for either enhancement or depletion mode operation. The meaning of these terms, and the differences between the different types, are explained below. The construction and operation of P channel enhancement mode devices is also described, since this type of device is the one most often used in digital arrays.

To fabricate a single device, passivated N type silicon is used as starting material. Holes are etched in the oxide and two P type regions are diffused in, using standard semiconductor processing techniques. These three stages in the process are illustrated in Figure 45.

A metallization pattern is constructed and connections are made to the metal contacting the two P regions and to the metal over the oxide between these two regions. The two P regions are called the source and the drain, and the in-between contact is called the gate. Figure 46 shows the result after the completion of this step.

2. Operation Principles

With no biases applied, the two P regions and the N type substrate or body form a pair of back-to-back diodes, with the result that the equivalent resistance between source and drain is very high. Figure 47 shows the situation when the gate is biased negatively with respect to the body. The negative gate potential creates a depletion region in the body below the gate, forcing the N type carriers away from the gate. For great enough biases, the polarity of the region between the source and drain is inverted to P type. In this situation, a P type channel has been induced below the gate and connects the source to the drain. Now the resistance from source to drain is much lower than in the case where no bias existed, and an appropriate source-to-drain bias will cause current to flow.

Since the conducting channel is induced by the application of gate bias, this is an enhancement mode MOS device. Since the channel, when it exists, is P type, this is a P channel enhancement mode transistor. In a depletion

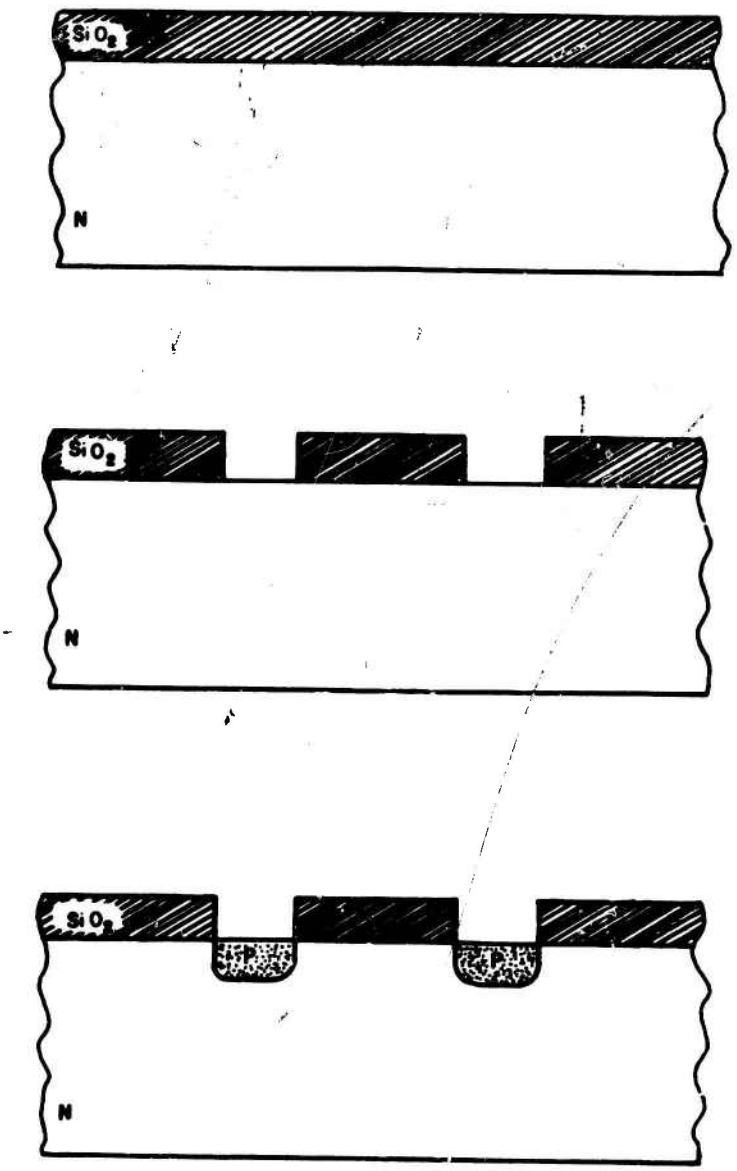


Figure 45. Three Stages of the Device Fabrication Process

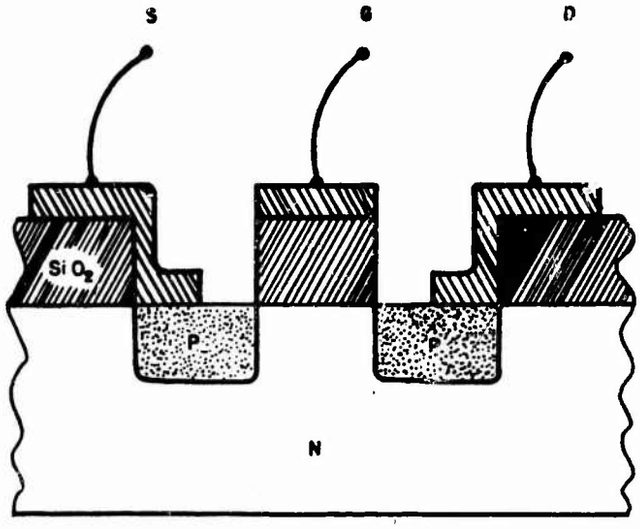


Figure 46. Device with Metal

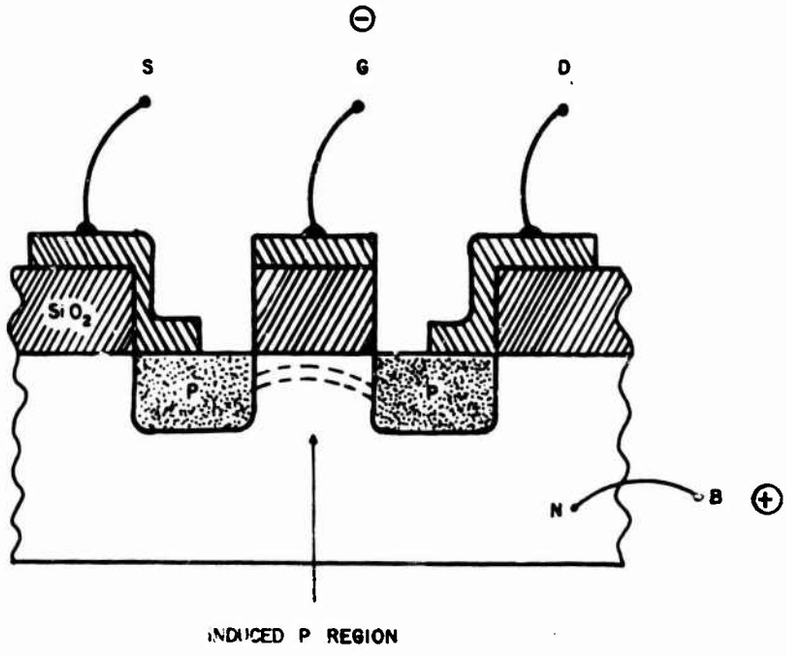


Figure 47. Device with Negative Bias

mode device, a channel is diffused into the body material just as the source and drain are. The channel is of the same type as the source and drain. In this situation, conduction from source to drain can occur in the absence of gate bias. Gate bias is introduced in such a polarity as to drive carriers out of the channel, changing its polarity to the opposite of that of the source and drain. For sufficient bias, the channel can be depleted of useful carriers; hence, the name depletion mode. The operation of N channel devices can be inferred from that of P channel devices by changing the polarities of the biases and of all the semiconductor regions.

Except where otherwise specified, all succeeding references to MOS devices in this appendix (as well as in the body of the report) relate to P channel enhancement mode devices.

MOS devices can be used as linear amplifiers, since the output conductance increasing with increasing gate bias. For digital applications, however, it is adequate to think of the device as a switchable resistor that is subjected to one of two discrete levels of gate bias; a level negative enough to turn the device on, or a level positive enough to turn the device off. The gate-to-source resistance will have a value of several megohms in the off state and a value of several kilohms in the on state.

The value of gate bias required to produce source-to-drain conduction is called the threshold voltage. The negative level must exceed this voltage by a sufficient margin to accommodate variations, from device to device, in threshold and in logic swing, and to provide some noise margin. Thus, the magnitude of the threshold voltage is one of the factors determining how large the logic swing must be.

MOS manufacturers are inclined to point to the threshold voltage as "built-in noise immunity". So it is, for signals at the more positive level. Noise immunity for the negative level, however, is determined by the amount by which the logic high exceeds the maximum threshold for devices in a circuit. Actually, a high threshold is something that manufacturers are trying hard to avoid. Less comment will exist about the advantages of high threshold voltage once manufacturers learn how to eliminate it.

B. CIRCUIT TECHNIQUES

One very attractive use of an MOS device is as a fixed resistor. Figure 48 illustrates this use. Here, the gate electrode is attached by the metallization pattern to the drain, causing the device to operate at a fixed on basis.

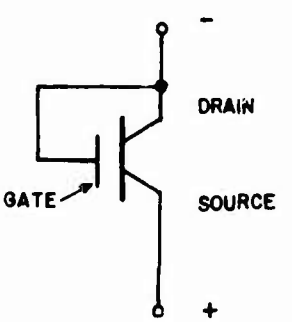


Figure 48. MOS Resistor

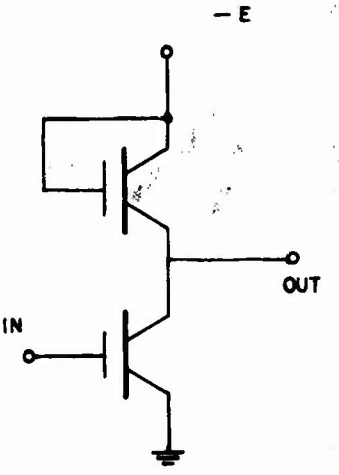
The value of such an MOS resistor can be in the neighborhood of 10K ohms. The value of this use of an MOS device lies in its efficient use of chip area. In a bipolar integrated circuit, resistors are diffused into the chip during one of the standard diffusion steps, usually the base diffusion. Since the depth and resistivity of the base diffusion is determined by the desired base width and base resistance of the transistors being made, the remaining variables affecting resistance value are the length and width of the resistor. The width will be made as small as possible, consistent with masking and etching capability. Then, the higher the resistance value desired, the longer the resistor must be. For resistors of appreciable value, the amount of area used up can be quite large. For example, a typical commercially available monolithic four-input DTL gate employs six diodes, two 2K ohm resistors, one 20K ohm resistor, and a transistor. The 20K ohm resistor takes up more than five times as much area as the transistor, and the three resistors together take up two-thirds of the total circuit area. A 20K ohm MOS resistor, on the other hand, can be fabricated in less area than that required for a single bipolar transistor.

Figure 49 shows a simple inverter and a three-input gate implemented with MOS devices. A simplified illustration of the layout and processing that could be used to implement the gate of Figure 49b is shown in Figure 50. The three P regions that are diffused in a single step are shown in Figure 50a. The lower two of these correspond to the sources and drains of the three input transistors in the gate. Since all the sources and all the drains are connected together, a single diffusion will suffice in each case. Furthermore, the drains of the input transistors are connected to the source of the load transistor; so the middle one of the three diffusions can be used as the source of the load transistor also. The top diffusion is the drain region for the load transistor. Figure 50b shows how metallization could be added to complete the circuit. The three input gates A, B and C are fabricated over the region between the main source and drain diffusions. The gate metal for the load transistor is deposited over the region between the main source and drain diffusions and also extends onto the top drain diffusion. This completes the connection from gate to drain for the load transistor. Two metallized areas for the output and ground connections complete the gate. This illustration represents a considerable oversimplification of the actual processing that would take place, but it does serve to illustrate the process and layout simplifications that result from characteristics of MOS construction.

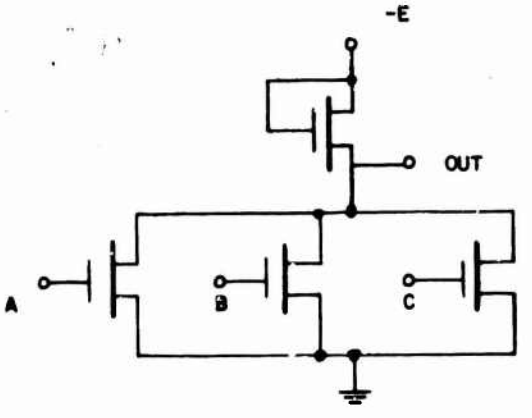
C. MOS CHARACTERISTICS

The characteristics of MOS that result in these simplifications deserve some discussion For one thing, an MOS device is a lateral type of structure. What is meant by this is that the source, gate, and drain are side by side on the substrate. A triple-diffused planar bipolar transistor, by contrast, has a vertical structure. The collector is diffused into the substrate, the base is diffused into the collector, and the emitter is diffused into the base. While it is at least theoretically possible to construct a bipolar gate analogous to the MOS gate of Figures 49 and 50, the emitters cannot be made in one piece, since each must be diffused into its corresponding base.

This lateral structure also contributes to the small size of MOS devices relative to bipolar transistors. There is a minimum dimension that any element of either type of device can have, based on process tolerances and masking capability. (This dimension is in the neighborhood of 0.0001 inch



a. Inverter

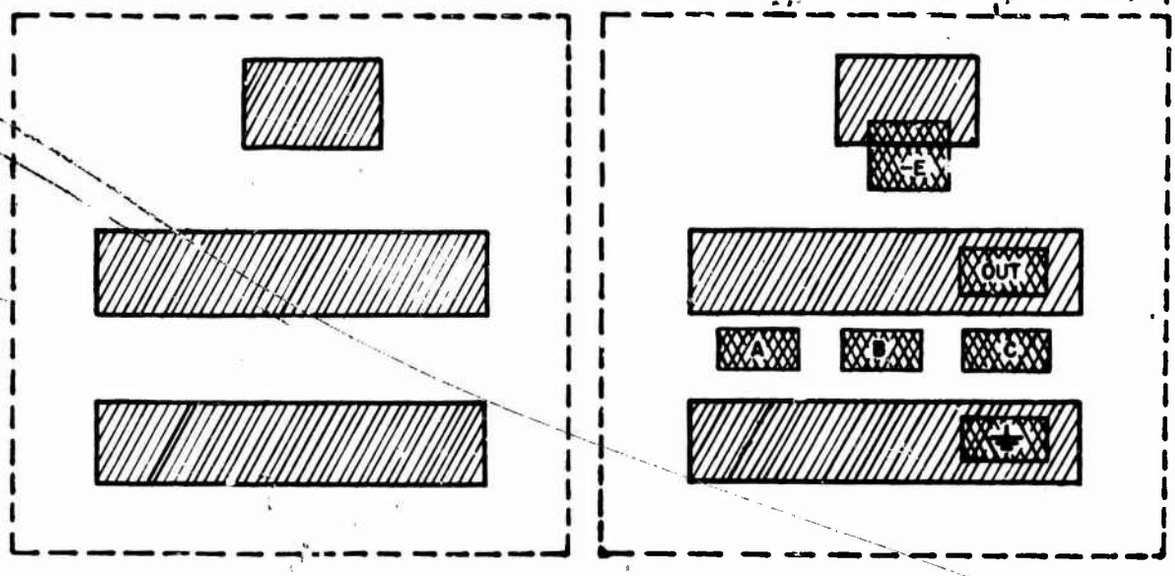


b. Three-input Gate

Figure 49. Inverter and Three-input Gate

for present technology.) Let this dimension be called one unit. From Figure 51, it can be seen that an MOS device that measures one unit by three units could be made. (For this ideal minimum-sized device, it is necessary for the gate metal to overlap the source and drain regions, as discussed later.) For the bipolar device, however, the collector region must be made large enough to contain the base region, the collector contact, and clearances between both of these and the edge of the collector. The base must be large enough to contain the base contact, the emitter, and clearances. The emitter and the base contact can be made one unit square. The resulting size for the bipolar device is nine units by five units, which covers fifteen times as much area as the MOS device.

Another factor that contributes to MOS devices being relatively small is the existence of automatic isolation between devices on a chip. An early step in the processing of a planar epitaxial NPN transistor in an integrated circuit



a. Diffusions b. Metallization

Figure 50. Gate Construction Steps

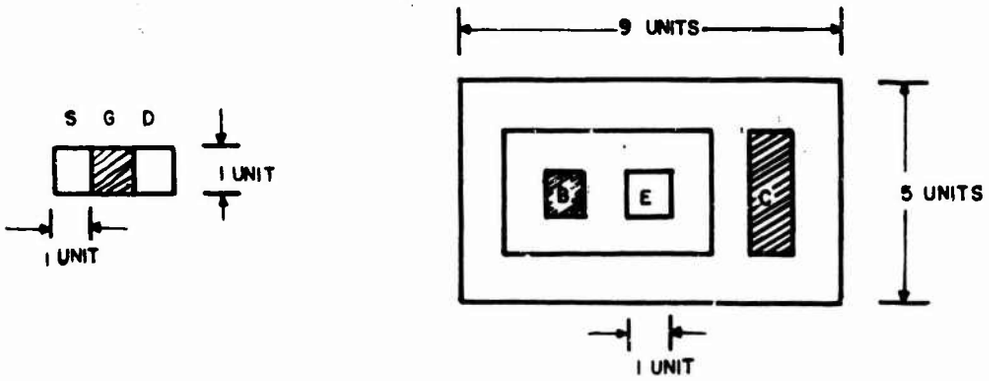


Figure 51. Relative Sizes of Bipolar and MOS Devices

is shown in Figure 52. The starting material has a layer of N type epitaxial material on a P type substrate. To define the collector regions for the transistors in the circuit, P type channels are diffused through the N material to the P substrate. The substrate and all the isolation channels are thus electrically connected and are biased to a voltage that is negative with respect to any collector on the chip. All collectors are then surrounded by back-biased diodes so that they are electrically isolated from each other. In the case of MOS devices, however, this isolation diffusion is not necessary. Figure 53 shows two adjacent MOS transistors on the same substrate. As long as the N type substrate is held at a positive voltage with respect to all of the source and drain regions, there will be no current flow between any of the diffused P

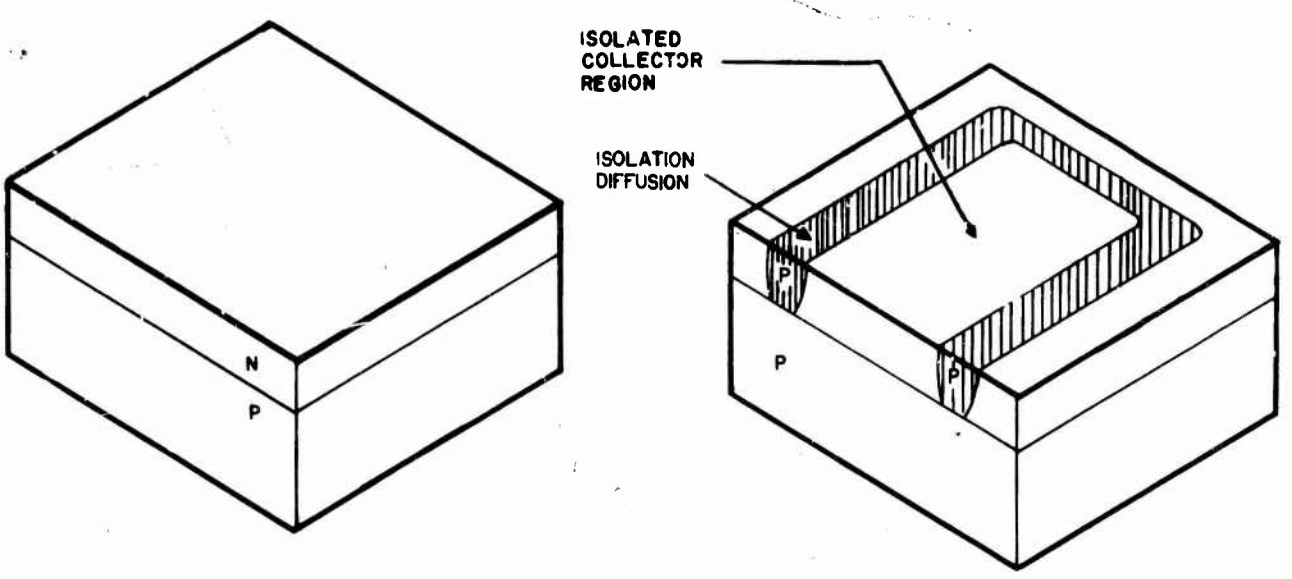


Figure 52. Bipolar Isolation Diffusion

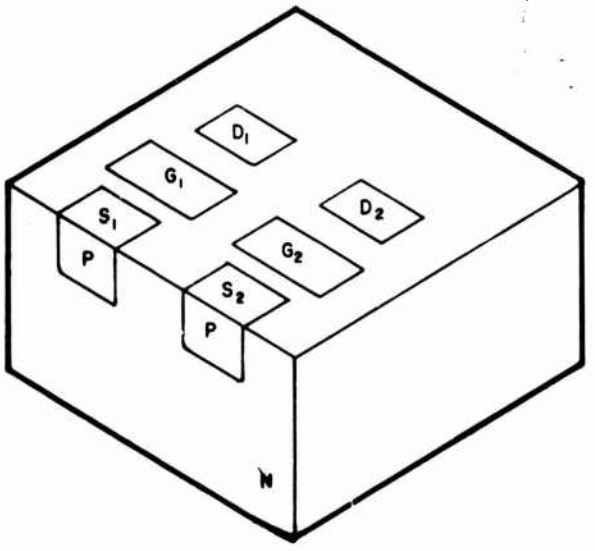


Figure 53. Isolated MOS Devices

regions unless a channel is induced between them. Since there is no gate electrode between S1 and S2, for example, no channel can be induced there and these regions are isolated from each other.

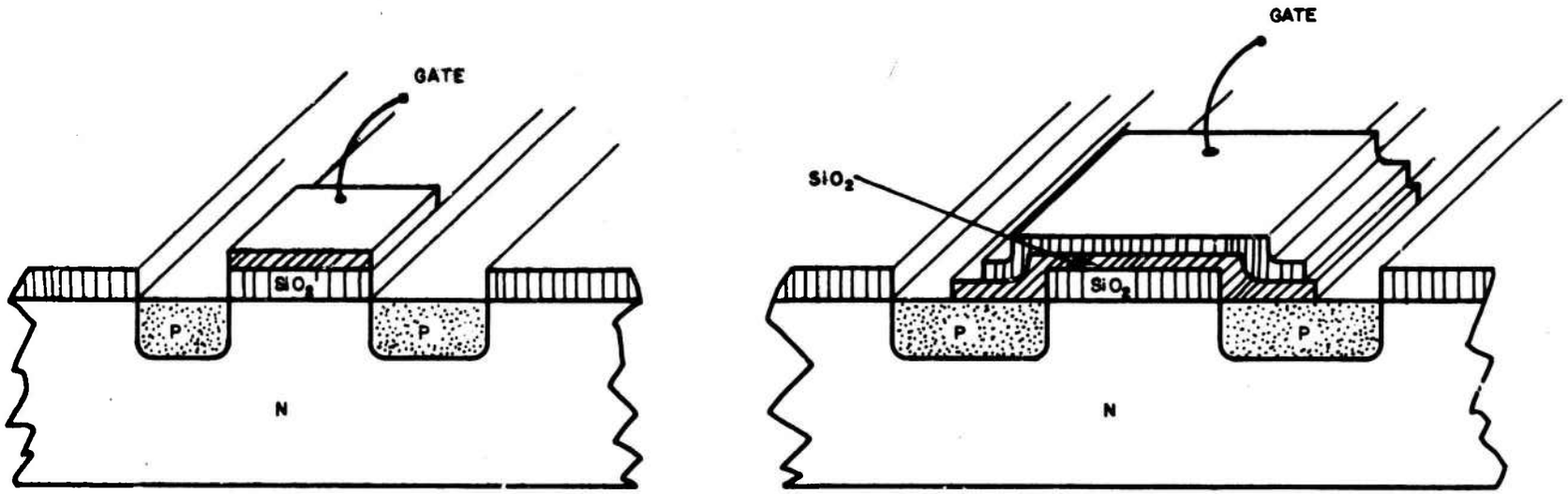
An interesting characteric of MOS transistors is that many types of digital circuits can be constructed with these devices alone. That is, circuits can be constructed entirely of MOS transistors, without any separately fabricated resistors and capacitors. Generally speaking, this is achieved by using MOS

load resistors as described above and by utilizing gate input capacitance where capacitance is required. For example, a dynamic shift register stage can be constructed with six MOS devices and no other elements except metallized interconnections. A static register stage requires nine transistors. In both cases, gate capacitance is used for storage while data is being clocked into a stage.

D. SPEED CAPABILITIES

The speed at which MOS devices can be operated is limited by the output impedance of each device and by the input capacitance of the device it drives. How low the output impedance can be is determined by the width of the device used as the output load (a wider channel implies lower on resistance) and by the allowable power dissipation in that device. Given that these two parameters are determined by other considerations, the device speed will be determined by input capacitances. In Figure 54a, the structure of an idealized gate is shown. The input capacitance of such a device is seen to be determined by the gate length (source to drain), the gate width, and the thickness and dielectric constant of the gate oxide. It is interesting to note that making the channel wider to lower the device on resistance will also increase input capacitance, so that switching speeds cannot be improved by this method. Making the channel length (source-to-drain distance) smaller, however, decreases both on resistance and input capacitance, so that the switching time is proportional to the square of the channel length. The desirable approach then is to make the channel as short as possible. Suppose that the channel length is made as short as is consistent with masking capability. Then the source-drain distance is the smallest dimension that can be worked with (referred to above as "one unit"), and the gate metal itself cannot be any narrower than this dimension. Since the metallization mask cannot be perfectly aligned with the diffusions, the gate metal must overlap either the source or the drain. This would result in a short from the gate to one of the other electrodes for the structure shown in Figure 54a. What is actually done is illustrated approximately by Figure 54b. After the source-drain diffusion, the material is re-oxidized and holes are etched to allow contacts to be made to the source and drain. The second layer of oxide is only shown over the gate region in Figure 54b. The contact holes are not etched all the way up to the gate, so that some insulating oxide exists over both the source and the drain adjacent to the gate region. The pattern for the gate metal in the metallization mask is then made wider than the gate region, to allow for misalignment. Thus, the gate must overlap source, drain, or both; but the second oxide is made wide enough to prevent shorting. The gate metal must be wide enough to accommodate tolerances in metallization width, tolerances in pattern size on the diffusion mask and on the metallization mask, and the alignment of the two masks. The result is that the gate metal length must be about three times the channel length.

The overlapping of the gate metal onto the source and the drain has a very pronounced effect on operating speed. Not only does the increased gate length contribute directly to increased input capacitance, but that portion of the gate which overlaps the drain is also subject to Miller effect. For example, suppose that a manufacturer wants to make an MOS register that will shift at 2 MHz. He must plan for the worst case of misalignment between



a. Idealized Gate Metal

b. Gate Overlap

Figure 54. Gate Construction

gate and gate metal, which could occur as shown in Figure 55. Let the channel length be L. Then the gate metal length must be 3L. Suppose the capacitance attributed to the portion of the metal over the channel (of length L) is C.

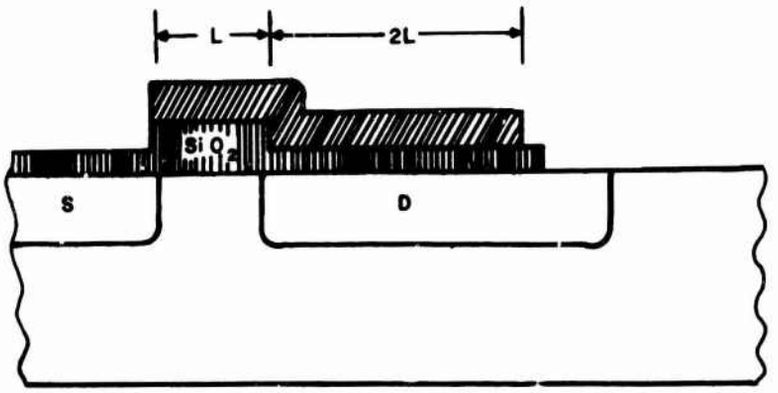


Figure 55. Gate Metal Overlapping Drain

The capacitance attributed to the portion of the metal over the drain must be 2C multiplied by the Miller effect. Suppose the gain G of the device is unity Then the total input capacitance is

$$C' = C + 2C(1 + G)$$
$$= C + 4C = 5C$$

What would happen if the manufacturer had some magic way of perfectly aligning the gate over the channel, so that the metal width only needed to be L? Then the total input capacitance would be

$$C'' = C = \frac{1}{5} C'$$

and, with other factors being the same, the shifting rate would be five times as high, or 10 MHz.

In other words, the only difference between a 2-MHz register and a 10-MHz register is the ability to perfectly align the gate. That difference turns out to be substantial. At present, the "magic method" that some manufacturers use to produce 10-MHz registers is to design the gate metal for zero overlap and to try to align the masks perfectly. Registers produced by this process either shift at 10 MHz or do not shift at all. Naturally, the process yield is low; but consumers who really need a 10-MHz register are willing to pay enough for one to compensate for the low yield. The individual who tries to buy a large number of these registers is in for a very depressing experience.

E. ADVANTAGES AND DISADVANTAGES

The advantages of MOS circuitry over bipolar are usually stated to be small size, low power consumption, simple processing, and low price. Actually, from the consumer's point of view, it would be better to say that the advantages of MOS circuitry are low price. This is true because small size and simple processing are advantages to the manufacturer, not the consumer, and as such contribute to low price. Even low power consumption falls into the low price category. Bipolar circuits would be made to have very low power consumption too, but only by using a great deal of silicon area or additional processing steps to create high-value resistors.

Most currently available MOS circuits have the disadvantage of having large logic swings. Furthermore, to obtain maximum operating speeds, it is necessary to employ a clock waveform that has greater amplitude than the logic signals. Logic amplitudes are 10 to 15 volts peak-to-peak and clock amplitudes are 15 to 25 volts peak-to-peak. These large swings are classified as a disadvantage because they aggravate system noise problems.

Probably the biggest disadvantage of currently available MOS circuits is the fact that their speed-power product (the product of gate delay and circuit power dissipation) is considerably greater than that of bipolar circuits. Speed-power product is used as a figure of merit for logic families. This use results from the fact that, for a given circuit, operating speed can be increased (within limits) by lowering resistor values. But doing so increases power consumption in such a way that the product of switching time and power remains about the same. So, speed-power product serves as a rough basis for comparing different types of logic circuits. The lower the speed-power product, the more attractive the circuit is. Speed-power product is roughly proportional to the square of the logic amplitude. Thus, a technology that requires a large logic swing will always have a greater speed-power product.

Another characteristic of present-day MOS technology, which is neither an advantage or a disadvantage, is that the major semiconductor manufacturers are just now getting into the business in a big way. The history of the semiconductor industry seems to indicate that a new technology really increases its attractiveness to the consumer much more rapidly after the major manufacturers enter the market place. (The situation during the introduction of integrated circuits a few years ago will bear this out.) This is at least partly due to the fact that the smaller suppliers can be profitable at a lower volume of business and do not require the high yields that are a prerequisite to volume production. So prices generally stay high, and process and technological breakthroughs are slower incoming. Large manufacturers, however, cannot afford to enter a new technology if " are only going to do five or ten million dollars of annual business in They require really high-volume production for the new technology to be profitable. Therefore, process improvements are needed to attain high yields; circuit and device improvements are needed to make the technology usable by many different types of customers; and low prices must be achieved to attract high-volume customers. MOS technology is in the early stages of the high-volume business now. A number of technological improvements that can be anticipated are discussed later in this appendix.

It appears likely that MOS circuits will never attain the speeds that bipolar circuits achieve. This is true not so much because of the higher speed power product of MOS, but because the areas where MOS has its strongest advantages work at cross purposes to achieving high speed. There are many ways that higher speeds can be achieved with MOS; the use of complementary devices, the incorporation of bipolar devices on MOS substrates, and the development of complementary MOS circuits are examples. These techniques are either available now or will be shortly. However, every one of them complicates the processing or takes up silicon area. The more of these approaches and others like them are developed to achieve higher speeds, the more complex the processing becomes, and the more area is required for a given function. In short, these methods work against the very things that make MOS attractive. It appears that manufacturers do not intend to compete with (their own) bipolar circuits but, rather, intend to make MOS complementary to bipolar technology.

F. EXPECTED ADVANCES

Manufacturers are working on a number of techniques for achieving two very desirable ends: level compatibility with bipolar logic circuits and higher speed. One technique aimed at producing bipolar compatibility is the development of silicon nitride gate insulating material to replace the silicon dioxide now used. Silicon nitride has a lower dielectric constant and yields a lower threshold voltage, making logic levels compatible with TTL logic circuits possible. The development of N channel devices is also being pursued. Because of the higher majority carrier mobility of N type material, it is anticipated that a three to five-fold speed increase will be obtained through this development. Both of these techniques are expected to be used in production in two or three years. Complementary circuits have both N and P channel devices on the same substrate. Circuits can thus have active pull-up and pull-down on outputs. Speed improvements of as much as ten to one are predicted for this technique.

Two techniques are now in development that contribute to both speed and bipolar compatibility. These are the silicon gate technique and the incorporation of bipolar devices on the MOS chip. Both of these advances are expected within the next year. The incorporation of bipolar devices is desirable because it provides low-impedance driving point capability, which MOS devices lack. These bipolar devices will first be incorporated to buffer chip outputs and later to drive clock lines, obviating the use of a clock larger than the logic swing.

Construction of a silicon gate is illustrated by Figure 56. In Figure 56a, the wafer is shown after the source and drain holes have been etched and before the diffusion step. The significant difference from standard processing is that the photoresist has been cleaned off before the diffusion step. The diffusion then takes place. The result is shown in Figure 56b. Due to the absence of the photoresist, a layer of polycrystalline silicon is grown on top of the oxide at the same time the source and drain are being diffused. This silicon layer is a conductor and can be used to replace the metal that would normally be used for the gate. The silicon can now be etched away from all areas except that over and immediately adjacent to the channel. Then a metal connection can be made to the adjacent area and the device is completed. The significant advantage of the silicon gate is that the conductor (silicon) over the gate region

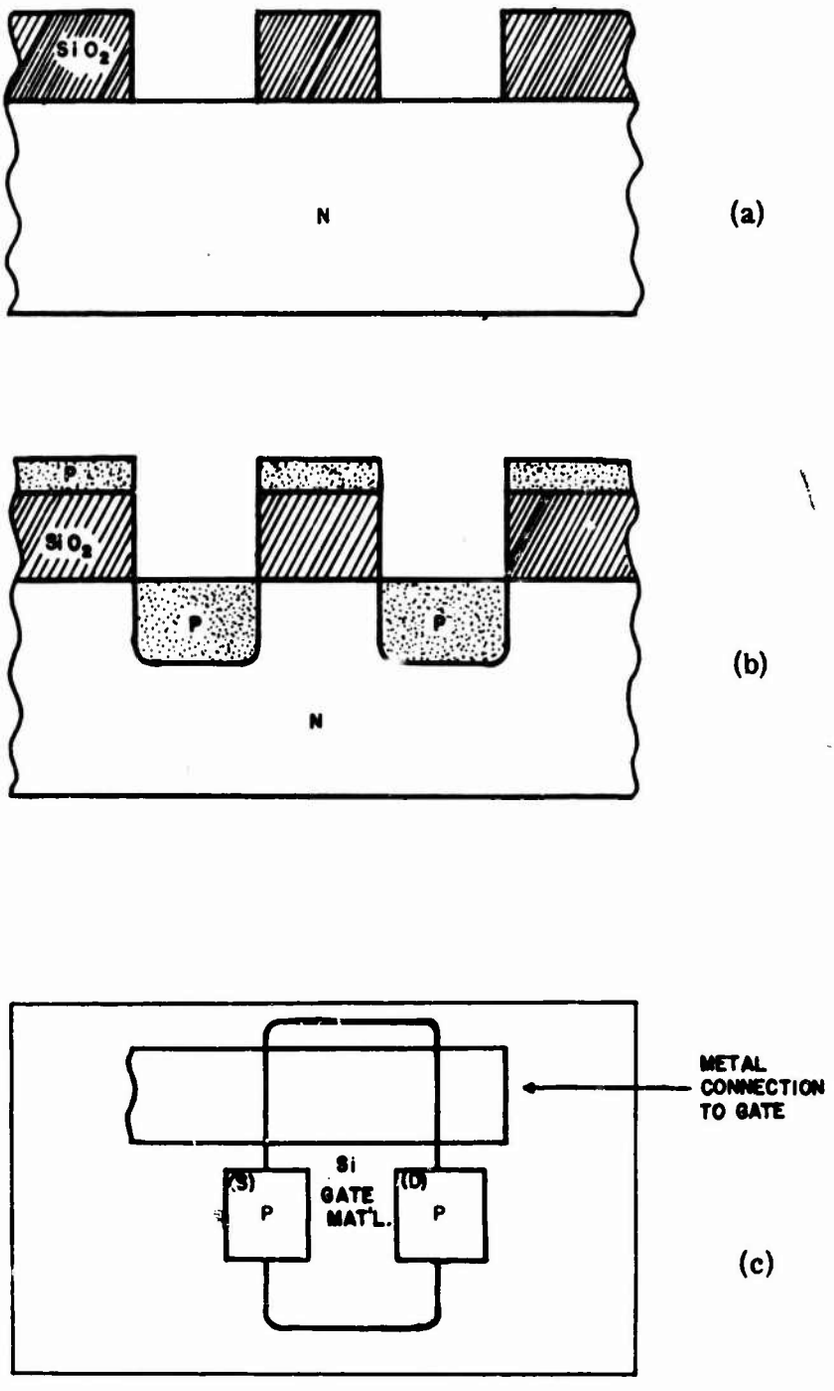


Figure 56. Silicon Gate Construction

is automatically aligned with the edges of the source and drain regions. A bonus advantage is that the replacement of the metal gate by silicon results in a lower work function at the gate-oxide interface and thereby produces a lower threshold voltage. Bipolar compatible circuits utilizing silicon gates and bipolar devices on the chip are expected to be in production within a year. Expected shifting rate for registers is 6 MHz.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) General Electric Company Apollo Systems P.O. Box 2500, Daytona Beach, Florida	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b GROUP -

3. REPORT TITLE

Study for Applying Computer-Generated Images to Visual Simulation

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Final Report (January 1969 - July 1969)

5. AUTHOR(S) (Last name, first name, initial)

Robert A. Schumacker, Brigitta Brand, Maurice G. Gilliland, Werner H. Sharp

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July 1969	131	2

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F33615-69-C-1280 b. PROJECT NO. 6114 c d.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AFHRL-TR-69-14

10. AVAILABILITY/LIMITATION NOTICES

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Air Force Human Resources Laboratory Aeronautical Systems Division Wright-Patterson AFB, Ohio 45433

13. ABSTRACT

This report describes the results of a system design study for applying digital image generation techniques to visual simulation for pilot training. The computer generated images are to provide out-the-window scenes for a flight simulator which is to be used for training Air Force pilots.

No existing visual system can provide all of the capabilities which are desired in a flight simulator. Digitally generated scenes do overcome many of the shortcomings associated with more conventional approaches but have had limited application because of the difficulty of computing enough image detail. The ability to generate images of more complex and realistic environments is closely tied to advances in digital device technology. The study assesses the impact of recent developments in this area on the design of an image generating system.

The conceptual design of an image generator is described. The principles of operation, the system configuration and operational characteristics are discussed. Several key problem areas are explored in depth. Feasible methods of implementation with presently available hardware are examined and an estimate of the hardware complexity is given.

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Visual Simulation						
Computer Generated Images						
Computed Displays						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

Unclassified

Security Classification